Structured Tracking for Safety, Security, and Privacy: Algorithms for Fusing Noisy Estimates from Sensor, Robot, and Camera Networks

Jeremy Ryan Schiff

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-104
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-104.html

July 23, 2009

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
23 JUL 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Structured Tracking for Safety, Security, and Privacy: Algorithms for Fusing Noisy Estimates from Sensor, Robot, and Camera Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Berkeley,Department of Electrical Engineering and Computer Sciences,Berkeley,CA,94720**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Emerging developments in the speed, size, and power requirements of processors coupled with networking advances, enable new applications for networks of sensors cameras, and robots. However, we live in a world lled with uncertainty and noise which a ects the sensors we use, the environments we model, and the objects we observe. In this dissertation, we de ne Structured Tracking, where we apply novel machine learning and inference techniques to leverage environmental and trackedobject structure. This approach improves accuracy and robustness while reducing computation. We focus on three application areas of societal bene t: safety, security, and privacy. We apply Belief Propagation (BP)[148] algorithms to sensor networks, and describe our modular framework for the more general Reweighted-BP formulation[203]. To improve safety, we track pallets in warehouses. We apply Particle Filtering[162] and model the cardiod-shaped response pattern of ultrasound between static beacons and mobile sensors to improve tracking accuracy by 11%. We also show using interdistance sensor readings, we can improve accuracy by 3-4x over the recent SMCL+R formulation[46], while being more likely to converge. We use a generalization of Particle Filtering, Nonparametric-BP (NBP)[183], which can model multi-modal and ringshaped distributions found in inter-distance tracking problems. We develop a novel tracking algorithm based on NBP to fuse dynamics and multi-hop inter-distance information that increases accuracy, reduces computation, and improves convergence. For security, we present a novel approach for intruder surveillance using a robotic camera controlled by binary motion sensors and use Particle Filtering to model intruder dynamics and environment geometry. We also present a localization-free approach to robot navigation using a distributed set of beacons, which emit a sequence of signals to direct a robot to the goal, modeling the robot's dynamics uncertainty, with up to 93.4% success rate. We introduce an approach to privacy for visual surveillance Respectful Cameras," that uses probabilistic Adaptive Boosting[68] to learn an environment-speci c 9-dimensional color model to track colored markers, which act as a proxy for each face. We integrate probabilistic Adaptive Boosting with Particle Filtering to improve robustness, and demonstrate a 2% false-negative rate.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**233**	

Structured Tracking for Safety, Security, and Privacy: Algorithms for Fusing Noisy Estimates from Sensor, Robot, and Camera Networks

by

Jeremy Ryan Schiff

B.S. (University of California, Berkeley) 2005

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

with a

Designated Emphasis in New Media

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Ken Goldberg, Chair
Professor Ruzena Bajcsy
Professor Joseph M. Hellerstein
Professor Deirdre Mulligan

Fall 2009

The dissertation of Jeremy Ryan Schiff is approved.

Chair	Date
	Date
	Date
	Date

University of California, Berkeley

Fall 2009

Structured Tracking for Safety, Security, and Privacy: Algorithms for Fusing Noisy Estimates from Sensor, Robot, and Camera Networks

by

Jeremy Ryan Schiff

Abstract

Structured Tracking for Safety, Security, and Privacy: Algorithms for Fusing Noisy Estimates from Sensor, Robot, and Camera Networks

by

Jeremy Ryan Schiff

Doctor of Philosophy in Computer Science

with a Designated Emphasis in New Media

University of California, Berkeley

Professor Ken Goldberg, Chair

Emerging developments in the speed, size, and power requirements of processors, coupled with networking advances, enable new applications for networks of sensors, cameras, and robots. However, we live in a world filled with uncertainty and noise, which affects the sensors we use, the environments we model, and the objects we observe. In this dissertation, we define Structured Tracking, where we apply novel machine learning and inference techniques to leverage environmental and tracked-object structure. This approach improves accuracy and robustness while reducing computation.

We focus on three application areas of societal benefit: safety, security, and privacy. We apply Belief Propagation (BP)[148] algorithms to sensor networks, and describe our modular framework for the more general Reweighted-BP formulation[203]. To improve safety, we track pallets in warehouses. We apply Particle Filtering[162] and model the cardiod-shaped response pattern of ultrasound between static beacons

and mobile sensors to improve tracking accuracy by 11%. We also show using inter-distance sensor readings, we can improve accuracy by 3-4x over the recent SMCL+R formulation[46], while being more likely to converge. We use a generalization of Particle Filtering, Nonparametric-BP (NBP)[183], which can model multi-modal and ring-shaped distributions found in inter-distance tracking problems. We develop a novel tracking algorithm based on NBP to fuse dynamics and multi-hop inter-distance information that increases accuracy, reduces computation, and improves convergence. For security, we present a novel approach for intruder surveillance using a robotic camera, controlled by binary motion sensors and use Particle Filtering to model intruder dynamics and environment geometry. We also present a localization-free approach to robot navigation using a distributed set of beacons, which emit a sequence of signals to direct a robot to the goal, modeling the robot's dynamics uncertainty, with up to 93.4% success rate. We introduce an approach to privacy for visual surveillance, "Respectful Cameras," that uses probabilistic Adaptive Boosting[68] to learn an environment-specific 9-dimensional color model to track colored markers, which act as a proxy for each face. We integrate probabilistic Adaptive Boosting with Particle Filtering to improve robustness, and demonstrate a 2% false-negative rate.

Professor Ken Goldberg
Dissertation Committee Chair

To those who support me on my journey: my family, my friends, and my mentors.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my advisor and friend, Ken Goldberg for his many years of mentorship. Whether it was the countless hours editing my papers, providing important research insights, suggesting concrete, actionable advice on better ways to present my research, or giving general advice about my career, you have always been there for me, and that is something I will never forget.

I would like to thank everyone at the UC Berkeley Automation Lab for the endless research conversations, suggestions, and feedback and support, particularly Ron Alterovitz, Ephrat Bitton, Vincent Duindam, Menasheh Fogel, Kris Hauser, Anand Kalkarni, Dezhen Song and Jijie Xu. In addition, I would like to thank my coauthors professors Max Meng, Deirdre K. Mulligan, Shankar Sastry, and Martin Wainwright, graduate students and post docs Dominic Antonelli, Nathan Burkhart, David Chu, Alex Dimakis, Siamak Faridani, Marci Meingast, Hongliang Ren, and Erik Sudderth and undergraduates Shervin Javdani and Danny Bazo.

I would like to thank my Qualifying Exam and Dissertation Committee members Joe Hellerstein, Sara McMains, Ruzena Bajcsy, and Deirdre Mulligan, for their suggestions about directions for future research and experiments, as well as ways for improving the presentation of my work.

I would also like to thank those who participated in the discussion and analysis of the societal ramifications of my research including Jennifer King, Abigail De Kosnik, Irene Chien, Kris Paulsen and Therese Tierney.

I would like to thank Panasonic for their donation of cameras, and NSF, AFOSR, TRUST, and Bayer Health Care for their funding.

Chapter 1

Introduction

There is an increasing societal need for security and safety. Advanced hardware is emerging with increased performance and reduced price. It is impractical for humans to monitor all of the information in real time. While there are many benefits to security and safety, automated monitoring systems can also encroach on personal privacy.

Technology can be part of the solution. In this dissertation, we develop new tracking algorithms for sensor networks, camera networks, and robot networks, that have the potential to provide some measure of privacy through masking and anonymization, while also providing the necessary monitoring for safety and security applications. These methods require new approaches to real-time tracking of people and objects. We introduce the concept of Structured Tracking, where we can leverage pre-existing structure or add additional application-specific structure to lower false positive/negative rates or reduce computational complexity. This dissertation advances the state of the art by proposing new algorithms that address three major application areas of societal benefit: safety, security, and privacy.

For more information including related news, updates, and live videos, please visit http://goldberg.berkeley.edu/jschiff.

1.1 Emergence of New Technologies

Over the last decade, there have been significant advances in hardware, distributed algorithms, probabilistic inference and machine learning techniques. These have provided the basis for the systems we designed and built as part of this dissertation that make our goals more feasible. We use these advances to address real-world problems, and in particular, design new algorithms to leverage technologies such as robot networks, and robotic cameras. Our new systems must work accurately and robustly in real-time. The challenge is that this real-time tracking must be able to address sensor noise, uncertain object dynamics, and environmental challenges such as changing lighting conditions.

Recent advances in sensors, processors, wireless networks, and distributed algorithms have resulted in new fields of research such as sensor, camera, and robot networks. Examples of improvements include greater sensor and transmission robustness, smaller size, and lower power usage. For instance, chips can be bought "off the shelf" from Dust Networks [1] that provide radio and robust wireless multihop networking (99.99% end-to end for a 50 node network [47]) are 12mmx12mm. These devices also have long battery lives due to efficient designs, allowing for 25 microamps when conserving energy, and below 200 microamps for typical heavy-duty applications. However, as these sensor networks are used to are log noisy data, frameworks for modeling and interpreting large amounts of probabilistic data have been a topic of active interest [71, 204].

Since 9/11/2001, new robotic pan, tilt, and zoom (PTZ) cameras have come on

the market with 42x zoom, built in web-servers for easy integration, and have dropped in price from $20k, to under $1k [2]. These robotic cameras are much more powerful than conventional static cameras. While static cameras have fixed resolution of the entire area, the user can direct a robotic camera to focus in and gather much more detailed information about a specific area of interest in the scene. However, our algorithms must address that the camera can move, and does not always view the scene from the same perspective.

Advances in probabilistic inference and machine learning algorithms, such as Particle Filtering [80], Adaptive Boosting [65], and Belief Propagation (BP) [148, 206], have dramatically increased the performance of approaches addressing learning and inference problems. These tools have shown promise in addressing detection, tracking, and classification problems in real-world environments with noisy sensing and dynamic environments. For instance, one of the most robust face-detection approaches (Viola and Jones) is based upon an Adaptive Boosting formulation. On 23 images with 149 faces, they achieved a detection rate of 77.8%, with 5 false positives on the MIT test set [198]. Many of our approaches, most specifically those using Particle Filtering, require the simulation of thousands of particles per second [168, 167]. Thus, most of our methods have only recently become computationally tractable due to the doubling of processor power every two years, as dictated by Moore's law.

1.2 Structured Tracking Design Principles

In this dissertation, we use a set of design principles that we refer to as Structured Tracking. Using these principles, we can improve reliability and robustness of our systems as well as reducing algorithmic complexity, making these approaches feasible for real-time tracking applications. These principles are:

1. Make use of application-specific structure
2. Insert additional structure
3. Use machine learning and inference techniques based on application-specific data

The concept of maximizing application-specific problem structure comes from research in fields such as Active Vision and Sensorless Manipulation. The field of Active Vision [106, 8, 187, 197] is based on insights that processing and interpreting of images does not need to be purely passive. By using streams of images coupled with camera movement, or adapting to more environmentally specific conditions such as modifying the iris to dynamically address brightness, we can provide more tractable formulations with more robust, computationally efficient solutions. Goldberg applies the idea of leveraging structure to the problem of orienting a part (up to symmetry), using no sensing at all: just controlling the distance and orientation of parallel jaw grippers [74]. Goldberg overcomes the need for sensing by using a priori knowledge of the part's geometry. In this dissertation, we introduce approaches which model problem structure:

- Use floor-plan layouts to constrain transitions of tracked objects in our models allowing for more accurate estimates
- Learn environment-specific lighting models for color-based tracking
- Model robot dynamics to improve static localization approaches
- Use spatial correlations between sensors learned for a specific environment

Researchers use the technique of Structured Lighting [165, 17], where structured signals of light that are known to the algorithms are emitted onto a surface. The

pattern of how this light is projected onto the surface provides more information to the system when trying to accomplishtasks such as producing depth maps. We use similar ideas, where we can introduce new structure to add additional constraints to make our problems more tractable and robust. In this dissertation, we describe approaches which introduce additional structure including:

- Put sensors on pallets to improve robustness for real-time tracking
- Place colored markers on people to improve robustness and reduce computational complexity for visual tracking
- Use active beacons to emit structured signals that reduces the computational complexity on the robot for autonomous navigation

Our approaches model application-specific problem structure using machine learning and probabilistic inference techniques. Machine learning techniques such as Adaptive Boosting [65] allow us to set the parameters of our models based on empirical observations. Probabilistic inference techniques such as Particle Filtering [162] and Belief Propagation [148] are flexible. Inference techniques allow us to model problem-specific structure including object dynamics, the environment, how sensor react to objects (observation models), and correlation between sensors, all to address tracking applications. Because inference algorithms reason using probability distributions, our approaches incorporate the uncertainty of each of our models into the estimate. These methods provide more than just the most likely estimate, but rather provides the entire distribution of our estimated state, which also gives information about the confidence in our estimates. Our models are mostly nonparametric, where distributions are represented as samples, rather than assuming parameterized distributions such as Gaussians. This allows our techniques to be more flexible, as we relax our assumptions about the underlying distributions, allowing us to address problems where

more standard, parameterized models cannot be used. For instance Gaussian models cannot be used to model the ring-shaped distributions found in our sparse pallet tracking problem described in Chapter 5. This flexibility also makes our approaches well suited for learning the distributions for our models, as we can reduce our assumptions about fitting a pre-conceived model.

1.3 Application Domains and Contributions

Each of our three application areas of safety, security, and privacy, use the concept of Structured Tracking to modify the problem formulations to assist in more accurate localization and tracking by reducing false positives/false negatives, or reducing the amount of necessary computation, making it feasible to our algorithms in real-time. The major contribution of this dissertation is the advancement of the state of the art in tracking, including both theoretical algorithmic improvements and practical applications of our results.

1.3.1 Safety: StatSense

In the fields of Robotics and Sensor Networks, we deal with distributed, noisy information. However, we can use spatial and temporal correlation to compensate for this noise and make our sensors more robust to false positives/negatives. In particular, we envision that sensor networks will be installed in "smart"-buildings for applications such as air-conditioning control or fire detection [209]. We explore how a sensor network with noisy temperature readings can use spatial correlation to compensate for faulty sensor readings, and particularly, experimented with inferring the temperature of locations that are not directly observed by any sensor.

Using the concepts of Structured Tracking, we introduce sensors that can commu-

nicate with one another, and share temperature readings. We also model and learn an environmentally specific spatial correlation of sensors for temperature prediction.

To our knowledge, we were the first to implement Belief Propagation[148] on a sensor network and provide real-world experimentation of our system. We demonstrated that in-network performance of BP is computationally and algorithmically viable. In particular, we show that BP is an ideal fit for distributed sensor, robot, and camera network applications due to the distributed nature of the formulation. BP also has the added benefit that because the algorithm provides distributions of the state we infer, rather than the most likely result, we can deduce the confidence of our estimates. We introduced and evaluated a modular architecture and framework to simplify the implementation by programmers of Reweighted Belief Propagation, a more general formulation of BP. We applied this framework to inferring the temperatures at unobserved locations in our lab, with on average of 3 degrees accuracy over a 45-degree temperature range, demonstrating the viability of BP for real-world applications.

1.3.2 Safety: Perceptive Pallets

There are many applications where we want to track many moving objects as they move around a confined, well-controlled space. For instance, at Bayer HealthCare Pharmaceuticals, pallets containing expensive and reactive chemicals travel around a warehouse. It is important to provide geometric guarantees about these chemicals such as maintaining a minimum distance between acids and bases. Failing to do so can result in $100,000 daily fines, $1,000,000 daily losses from plant closures, or fires that will destroy the warehouse and can expose nearby neighborhoods to toxic chemicals.

In the Perceptive Pallets project, we explore how to track pallets in a warehouse.

We experiment with two different architectures. In the first, we use a dense-beacon architecture, and the pallets are tracked only using sensor-readings between beacons and pallets. The beacons must be dense because the inter-distance sensors have limited range, and the pallets must receive information from at least three beacons to localize themselves. In the sparse-beacon architecture, we fuse information in a more distributed fashion, requiring only three beacons in the entire deployment, but also using sensor-readings between pallets.

Using the concepts of Structured Tracking, for both architectures, we attach a sensor to each pallet to remove the problem of identification and improve the robustness of tracking. We use models of the dynamics to fuse information between timesteps, and introduce beacons to reduce the complexity of our tracking algorithms. As our two architectures vary the amount of structure we introduce into the problem by having different assumptions about beacons, we are able to see how this structure affects our algorithms. With a dense-beacon architecture, which may be infeasible in some applications, we can use simpler, more computationally efficient algorithms. With a sparse-beacon architecture, we need to use inter-sensor distance information, requiring more sophisticated algorithms to compensate for noisy location estimates of neighboring pallets, making the algorithms slower. Using noisy estimates of neighbor locations, rather than precise beacon locations, also reduces the accuracy, as error will propagate from nodes closer to beacons to nodes further away.

In Dense Perceptive Pallets, we leverage insights about the asymmetric, cardiod-shaped response pattern of ultrasound-based distance information to improve tracking accuracy. By modeling how the distance information provided by time difference of arrival methods (TDOA) actually provides information about orientation as well as position, we are able to formulate a Particle Filter that improves accuracy by 11% in our deployment.

In Sparse Perceptive Pallets, we develop a novel algorithm, based on the non-parametric Belief Propagation framework, for the tracking of distributed sensors with inter-distance readings. This allows us to incorporate information in a purely distributed fashion, using limited range inter-distance readings between neighboring sensors to localize and track all the sensors simultaneously. The use of a non-parametric representation also allows us to model the noisy ring-shaped and multi-modal distributions intrinsic to inter-distance localization and tracking problems. We use this framework to explicitly model inter-distance sensor error, the noisy robot dynamics, and noisy network connectivity. We describe a novel tracking algorithm, extending the localization approach of Ihler et al.[86] that intelligently decomposes the dynamics model, allowing us to fuse the robot dynamics with information from robots that are multiple hops away (in the graph where robots that directly receive inter-distance information from one another are one-hop neighbors). This allows us to use dynamics information to improve accuracy, as well as reduce computation and the number of transmitted messages when compared to a localization approaches such as Ihler et al.'s which treat each timestep independently. In addition, our inference algorithm models additional constraints given by the dynamics allowing for improved robustness when the sensors receive inter-distance readings from fewer than three neighbors. We show a 3-4x improvement in accuracy over the recent SMCL+R algorithm[46], and also demonstrate that our approach is more robust, as we do not assume the unrealistic assumption of a unit-disk connectivity model[207]. In such a model, all inter-distance readings are assumed to be received within a threshold distance, and none are received beyond it. In contrast, we accommodate noisy connectivity models, which we model formally using probability distributions.

1.3.3 Security: SNARES

In our project on Sensor Networks for Robust Environment Security (SNARES), we wish to monitor an environment to secure it from an intruder using a robotic camera. The camera is controlled by incorporating information from a network of cheap, binary passive infrared sensors.

Using concepts from Structured Tracking, we model the dynamics of the person, and use a priori models of the room's geometry to improve the accuracy of our transition models by preventing infeasible transitions such as traveling through walls. Because we explicitly place the sensors within the space at known locations, we use the spatial correlation between the sensors that trigger (and those nearby that do not) to track the intruder.

The SNARES approach is a novel approach to security surveillance. In contrast to the typical approach of observing a space with many static cameras, we use a single robotic camera, which is controlled by a sensor network, and a Particle Filtering-based tracking algorithm. We develop a novel mechanism for describing the spatial response pattern of binary sensors, which allows us to characterize them once, and then when deploying, automatically incorporate the information from each sensor's current status (of triggering or not) to track objects in the world-coordinate frame. We also describe how to model a sensory refraction period, where after triggering, a sensor is blind for a specified period, and during that time will not trigger again.

1.3.4 Security: Actuator Networks

In our Actuator Network project, we propose and evaluate a novel method for robotic navigation. Instead of the robot localizing itself, and using that information to determine the robot's next move, we place a network of actuators that act as beacons

in the environment, which guide the robot by actively changing their emitted fields. By adding beacons to the environment, we reduce the complexity of technologies for sensing and associated algorithms needed by the robot while also removing the need for the robot to communicate with the beacons. The robot's navigation sensor is limited to the simplistic sensing of the local gradient it observes from the actuators emitting a signal, such as light. The robot follows gradient descent of the sum of emitted fields, and by varying the fields, the beacons direct the robot along the path with greatest probability of reaching the goal. The control algorithm uses an open-loop planner that does not observe the robot at intermediate timesteps.

We can use this framework to control beacons to covertly guide a robot. Because the robot passively observes the sum of signals from the actuated beacons, rather than communicating directly, it is difficult for adversaries to monitor the signals to determine the path of the robot determined by the beacons. While we didn't experiment with the application and our current formulation only addresses two-dimensional navigation problems, the Actuator Network framework could be extended to guide a robotic fly [181] to a location within a space for covert surveillance.

Using the concepts of Structured Tracking, we place beacons in the space to reduce the hardware and algorithmic complexity required for robot navigation, rather than relying on navigation by detecting unique landmarks. By modeling the robot's movement, and structuring the schedule of the signals emitted by the beacons, we are able to reduce the computation required by the robot for navigation. Also, we use our models of robot dynamics, given different emitted signal configurations, to maximize the probability of successfully directing the robot to a goal location.

Because our approach uses beacons to guide the robot to a goal without observing it, we developed an open-loop planner to determine the set of fields that are most likely to result in the robot successfully navigating to the goal, modeling the robot's

noisy dynamics. In other navigation approaches, which also model the problem as gradient descent on a potential field, the fields are purely virtual, allowing them to be emitted from any location with any shape. Because our planner is limited to physical fields produced by actuators, we provide a different, novel solution whereby robots travel along waypoints to the goal locations, determined by the incenters of triangles formed by any three actuators. We achieve up to a 93.4% success rate in our simulations.

1.3.5 Privacy: Respectful Cameras

In Respectful Cameras, our goal is to obscure faces of people in video in real-time to provide the observed people with some measure of privacy through anonymity. To more robustly address the face-detection problem, instead of directly determining the location of each person or face, we use the novel approach of placing colored markers on the people we wish to track. This helps compensate for limitations of current facial/person detection approaches not having sufficient robustness for this privacy problem. In particular, these approaches fail when addressing issues such as arbitrary facial/body orientation, dynamic lighting, and partial obstructions. This also compensates for background subtraction methods, which are difficult to apply to scenes observed with robotic cameras.

Using concepts from Structured Tracking, we add additional structure to the conventional face/person detection problem by detecting a marker instead. Using this structure with models of people dynamics allows us to decrease the false positive and false negative rates, especially when subject to environmental changes and partial obstructions, while simplifying the algorithms allowing us to process the data in real-time.

We provide an interface allowing a user to quickly train the system on a specific

marker's colors, as determined by the lighting for a specific environment. Our results suggest that by using a 9-dimensional color space of RGB, HSV, and LAB, and performing machine learning of each color-space independently, we can provide better tracking accuracy than using only one-color space. We use a reformulation of the Adaptive Boosting[199, 145] machine learning algorithm, providing the probability that a pixel color corresponds to our marker (versus the background), so that it can be used with a Particle Filter [162] for tracking. We extend our approach to track multiple-objects simultaneously. We achieved a 2% false-negative rate in our deployment at the CITRIS construction site at UC Berkeley.

Chapter 2

Related Work

Many researchers have investigated how to accurately and robustly track objects, but they often assume this tracking is purely passive, being unable to interact with or modify the tracked object or the environment. To perform tracking we must first solve the sub-problem of localization. Localization is the task of identifying the location of an object at a single instant in time, using the information available only at that timestep. Tracking, also referred to as dynamic localization, utilizes previous and/or future location estimates and a dynamics model to make more informed estimates of the current objects' location.

2.1 Tracking Sensors Attached to Objects

2.1.1 Primitives for Tracking Sensors Attached to Objects

If the application domain permits the attachment of a device, such as a sensor node, to the object being localized, then there are three main approaches to approximate distances information to the object. Once we select one technique, we can apply tracking algorithm to compensate for sensor error and use information about robot

dynamics and environment structure. These techniques are frequently discussed in the sensor network literature, for tracking sensor motes. Motes are small devices equipped with sensors, memory, processors, and wireless communication hardware [70].

RSSI, or Received Signal Strength Indication is the received signal strength of the wireless transmission. Lorincz and Welsh use an approach where they look at the RSSI at multiple frequencies, and have a calibrated map of what the RSSI should be on each frequency at many locations in the space. Thus, they can lookup, for a given set of RSSI frequencies, the most likely location of the mote [121].

TDOA, or Time Difference of Arrival, is the localization system for systems such as cricket motes [154]. This approach works by sending a radio and ultra-sound pulse at the same time, and using the difference in arrival time to compute distance. While most conventional TDOA approaches compare radio to ultrasound, a company Ubisense recently introduced a TDOA-based approach that uses ultra-wideband which they claim has better performance due to reduced multi-path effects [4]. The claim 15 cm accuracy, but their technology costs $14,950 for a 9-tag research kit.

RIP, or Radio Interferometric Positioning [123, 107], uses motes broadcasting radio at slightly different frequencies. The resulting phase offsets as observed by receiving motes are used to determine the inter-mote distance. Initially, they showed 3 cm accuracy with a range of up to 160 meters between motes, without addressing multi-path effects. They extended this to do tracking, running at a rate of 1 sample every 4 seconds.

2.1.2 Algorithms for Tracking with Sensors Attached to Objects

There is a large body of literature related to algorithms for tracking with sensors, which are placed on objects to be tracked, as we do in our Perceptive Pallets applications (see Chapters 4 and 5). Thus, the problem formulation is for the sensors to localize themselves, relative to the environment. Related Problems include simultaneous localization and mapping (SLAM), simultaneous localization and tracking (SLAT), control of robotic swarms, and robot/sensor network localization.

In the classic SLAM problem, there is a single moving robot that produces a map of the environment while simultaneously determining its location. Several recent approaches allow SLAM problems to be efficiently solved with as many as 10^8 static features where distributions are modeled as multivariate Gaussians. Many of these methods are based on inference in an appropriate graphical model; for an overview, see [191]. SLAM has also been extended to domains with multiple robots, often under the assumption that the initial locations of the robots are known [30, 195]. In these approaches, robots share and localize using a global map, rather than through distributed observations of other robots as in our approach. Thrun and Liu [192] investigate merging multiple maps when initial locations of robots are unknown, and the landmarks are not uniquely identifiable. Howard [82] explores map merging when robots do not know their initial locations.

The SLAT problem is a variant of SLAM in which static "robots", which are typically nodes in a sensor network, use distance and bearing readings from a moving robot to localize themselves. The approach of Taylor et al. [189] avoids representing ring-shaped or multi-modal posterior distributions by assuming sensors roughly know their initial locations, and processing sets of distance measurements in batch. Funiak

et al. [69] propose a more complex method focused on SLAT problems in camera networks.

In the distributed control community, methods have been proposed for controlling a swarm of robots to maintain a specified formation [216], or using several mobile robots to localize a target [215]. In these approaches, sensors are assumed to observe true locations plus Gaussian noise, resulting in posterior distributions which (unlike those arising in inter-distance tracking) are well approximated as Gaussian. Like these methods, however, we also seek to develop effective distributed algorithms for multiple robots.

There has been a great deal of research on distance-based localization in sensor networks; for a summary, see [207]. Most of these rely on technologies for measuring inter-distance information described in Section 2.1.1. In these approaches, sensors communicate distance information between one another, or with beacons, rather than being used to interrogate an unknown environment. Localization techniques from this literature can be applied at each timestep to determine the location of each robot, to approximate tracking, but ignore dynamics. To address static localization problems, researchers have applied techniques such as multidimensional scaling (MDS), in both centralized [171] and distributed [92] implementations. Many localization methods produce global solutions by incrementally stiching local sensor maps together [92], for instance by recursively localizing new sensors with respect to previously localized sensors [163]. Other approaches solve more global inference problems, computing location estimates via iterative least squares [43, 136] or approximate marginal distributions via NBP [88].

In some applications of multi-robot tracking, a sufficient number of beacon nodes (at least three) can be directly observed by all robots at all times [179, 158]. Other approaches assume sensors measure both orientation and distance, allowing for simpli-

fying Gaussian assumptions [161]. Previous approaches to the inter-distance tracking problem include the work of Howard et al. [83], which formulates inter-distance tracking as numerical optimization; Park et al. [73], which solves a trilateration problem using quadratic approximations of the motion of robots relative to one another; and Dil et al. [46], which uses assumptions about maximum transmission radii to apply Particle Filters [12].

There have also been research validating that applying tracking methods can improve localization accuracy. Fogel et al. explore the use of a particle-filtered approach for tracking, explicitly modeling angular dependency of an ultrasound-based time difference of arrival (TDOA) localization algorithm of the MIT Cricket mote [61]. Smith et. al used an Extended Kalman Filter (EKF) to improve the tracking performance of Cricket motes [178].

2.2 Tracking Objects Via External Sensors

There are other research problems associated with tracking objects using external sensors which observe the objects that should be tracked. In contrast to placing sensors on the objects to ease tracking, external sensors are placed statically in the environment, and are used to observe nearby objects. Some of these sensors like Passive Infrared (PIR) and Cameras are passive, and just collect incoming data from the world, while others, like Laser Range Finders emit a signal to collect information. As we cannot place sensors on the objects to track them, these problems are often posed where the tracked objects are adversarial, and seek to avoid detection. We explore this type of problem in Chapter 6.

For sensor-network applications, Oh et al. [143] explores the problem of efficiently tracking multiple objects in a sensor network using a MCMC approach, with hier-

archical fusion of sensor information (as opposed to purely distributed, or purely centralized). Oh et al. explore a tracking of multiple objects with binary sensors, using PIR in experiments. Other examples range from locating a stationary intruder in an unknown room [52] to pursuit-evasion games, where a pursuing robot hunts a mobile evader that intelligently works to avoid capture. Examples and surveys can be found in [90], where the pursuer has a line of sight optical sensor, and [15], where the pursuer must track the evader while avoiding being seen.

While these types of problem require less structure, as we do not have to place a sensor on the object, there are ambiguities as to the identities of multiple objects moving through the space, which require additional algorithms to resolve. The research community refers to this issue as the data association problem, where the goal is to associate object locations at one timestep, with object locations at another, when no identity information is present. This results in uniquely identifying the paths of a varying number of objects. One approach for this problem is to use dynamics to resolve the ambiguities. An early approach to this problem is multiple hypothesis tracking [155], where a set of hypothesis are used from all previous timesteps, and integrated with the information at the current timestep, to generate a new set of hypotheses. The hypothesis with the greatest posterior probability is assumed to be the objects' true location. This has been shown to be an effective solution, however, the number of hypothesis grows exponentially as time progresses. As a result, new approaches such as using Markov Chain Monte Carlo (MCMC) [142] have seen better performance.

2.2.1 Vision-based Tracking

If the application requires passive observation, rather than attaching something to the objects we wish to track, we can perform tracking with a camera network.

The first step to tracking with a camera network is using object detection, where we determine all of the instances of a specific object in an image. If we assume, as is often the case, that the cameras are calibrated such that we know the camera position and orientation, once we have detected the object in at least two cameras, we can localize it. Due to the frequency of applications involving humans, there are specialized algorithms for face [27, 196, 198, 26], and people detection [211] which we can use for people tracking.

There are two major approaches to object detection: features that are pixel-based and features that are region-based. In pixel-based, pixels are classified as corresponding to the object or not, and then clustered together. In region-based, groups of pixels are classified together, and no clustering is needed.

One common pixel-based method is the use of skin color as a mechanism for face or people detection. For example, Lin et al. verify if each pixel satisfies a priori color constraints [119], another approach uses machine learning [51] to model face color. Others use multivariate Gaussians [213, 10] and Gaussians Mixture Models [76] to represent the face colors. While each pixel in images from cameras are typically represented using a Red, Green, and Blue (RGB) colorspace, most approaches transform the image to another color space such as Hue, Saturation, Value (HSV), LAB or YCrCB, but in contrast to our vision tracking work, few use more than a single colorspace. These color-spaces are often chosen for explicit modeling advantages, for instance HSV seeks to decouple color, namely hue and saturation, from brightness, described by the value component which has the goal of making models more robust to changing brightness. It is also common to use pixel color as a pre-processing step to prune out image regions before using a more powerful intensity-based face detector [213]. As these systems evolve and are used in industry, it is important that these systems are trained over a wide variety of races and skin-tones so that the system does not work for some, and not for others.

Avidan describes the use of a pixel-based method [13] where each pixel's initial feature vector contains the RGB values, as well as two histograms of oriented gradients similar to those used in Scale Invariant Feature Transform (SIFT) features [122]. These SIFT features are commonly used for problems such as the correspondence between images or in image classification.

Typical region-based approaches explore applying features like Haar wavelets which generate statistics about regions of pixel [199, 14], and therefore do not need a clustering step.

Approaches to object detection frequently employ statistical classification methods including AdaBoost [199, 145], Neural Networks [56], and Support Vector Machines (SVMs) [146] to learn models for how the features statistics correspond to the objects we wish to detect.

Object detection methods can be used with temporal models to perform camera-based tracking. Most frequently, these apply to face [48, 149] and people tracking [144, 212] using image processing and vision systems. These techniques have also been extended to fuse detections across multiple views in a decentralized fashion using a camera network [205]. Unfortunately, these methods have difficulty detecting and tracking in real-time for domains with partial occlusions, arbitrary pose, and changing lighting conditions [105, 214].

Alternatively, background subtraction methods, based on formulations such as Gaussian mixture models [114], can be applied to detect foreground objects. These methods generally assume that all moving objects are objects of interest, and are generally not designed to disambiguate between different moving objects. However, there has been limited work that uses object models and filtering as a post-processing step to disambiguate between objects and compensate for occlusions [169, 89, 210, 132]. Many background subtraction methods will eventually determine that a stationary

object, is part of the background, which will cause the system to lose the location of the tracked objects. For domains where always tracking the object is of utmost importance, like privacy systems, where misplacing a person's location for one frame can result in the loss of anonymity for many future frames, these limitations can be significant. Also, background subtraction techniques struggle with domains using robotic cameras, as the system will have to build up a background-model for the new perspective, and in the meantime, newly observed moving objects may not be identified.

Some researchers, like Shet et. al [173], have explored how object detection can be simplified by placing a unique near-infrared tags on each object. Others reduce the computational requirements and improve robustness of object detection with visual markers. Zhang et al. compared many of these methods [221]. Kohtake et al. applied visual markers to reduce algorithmic complexity and improve robustness of object classification to ease the user interaction problem of transferring data between digital devices by pointing and selecting physical objects via an "infostick" [103].

Because of the limitations of background subtraction methods, and difficulties with general object detection, our approach for preserving visual privacy described in Chapter 8 makes the system's tracking more robust by adding the additional structure of a colored marker worn by those we wish to track. We use a form of AdaBoost [65] to learn a environment-specific color model subject to the environment's varying lighting conditions. AdaBoost is a supervised learning approach that creates a strong statistical classifier from labeled data and a set of weak hypotheses, which poorly classify the labeled data. Rather than conventional AdaBoost that provides a binary label, we use Probabilistic AdaBoost [68, 164], which provides the probability of an input's label that we use in our tracking algorithms based on Particle Filtering. In contrast to other approaches, we use multiple color spaces simultaneously as our feature vector, providing additional flexibility for improved learning robustness.

The research closest to Respectful Cameras is the work of Okuma et al. [144] and Lei et al. [116], who also use a probabilistic AdaBoost formulation with Particle Filtering. However, both assume a classifier per tracked-object (region-based), rather than classifier per-pixel. As our markers use pixel-based color, we do not need to classify at multiple scales, and we can explicitly model shape to help with robustness to partial obstructions. Okuma's group applies their approach of dynamic weighting between a Particle Filter and an AdaBoost Object Detector to tracking hockey players. Rather than weighting, our approach directly integrates AdaBoost into the Particle Filter's observation model. Lei et al. uses a similar approach to ours, and performs face and car tracking. However, unlike Lei, our formulation can track multiple objects simultaneously.

2.3 Navigation

Once we are able to track objects, a natural next step is to explore robot navigation. Many navigation problems are posed as determining a robot's path by a potential field.

There is a long history of investigation into the characteristics of potential fields induced by physical phenomena and how objects are affected by these fields. Some of the earliest work, related to the study of gravitational, electric and magnetic fields, as well as topological properties of such fields, was pioneered by Newton, Gauss, Laplace, Lagrange, Faraday, and Maxwell [98, 125].

Potential functions for robotic navigation have been studied extensively as tools for determining a virtual field which a robotic element can follow to a goal while avoiding obstacles. See Choset et al. [36] for an in-depth discussion. Khatib proposed a model where a goal location is represented as an attractor, obstacles are represented

as repulsers, and the overall field is determined by the superposition of these fields [99, 100]. While these fields are typically referred to as potential fields, they are actually treated as vector fields determining the movement vector for the robot at any location in the space. While moving, the robot performs simple gradient descent on this space, and the objective is to define attractive and repulsive fields such that the robot will always end up at a local minimum created at the goal. Several extensions exist to this classical approach, for example to prevent the robot from getting stuck at a local minimum [39] and to deal with moving obstacles [140, 100].

Rimon and Koditschek addressed this problem by determining the attractive and repulsive potential functions necessary to guarantee unique minima [156]. They accomplished this by defining functions over a "sphere world" where the entire space and all obstacles were restricted to be n-dimensional spheres. They discovered a mapping from this solution to other types of worlds such as "star-shaped worlds", allowing for a general solution to this problem. Connolly et al. [39] use harmonic functions in the repulsive and attractive functions to avoid local minima.

The concept of distributed potential fields has also been studied in various contexts. Li et al. [118] examined how to directly apply potential functions in a distributed fashion over sensor networks, focusing on formulating the algorithm in a distributed manner. Pimenta et al. [150] addressed the robot navigation problem by defining force vectors at the nodes of a graph. Finally, research in distributed manipulation has examined how to leverage many actuators to perform coordinated manipulation, focusing primarily on the use of vibratory fields to place and orient parts [23, 20, 184, 108, 201].

To actively construct potential fields to steer a mobile robot subject to uncertainty in motion and sensing, we build on previous results in motion planning under uncertainty [111]. Motion planners using grid-based numerical methods and geometric

analysis have been applied to robots with motion and sensing uncertainty using cost-based objectives and worst-case analysis [25, 112, 111]. Markov Decision Processes have been applied to motion planning with uncertainty in motion, but these methods generally require direct observation of the state as the robot moves [44, 57, 9, 111]. To address motion uncertainty, Alterovitz et al. introduced the Stochastic Motion Roadmap (SMR), a sampling-based method that explicitly considers models of motion uncertainty to compute actions that maximize the probability that a robot will reach an end location [9]. Lazanas and Latombe proposed a landmark-based approach to robot navigation in which the robot has improved sensing and actuation inside landmark regions, reducing the complexity of the motion planning problem to moving between these regions [113].

Our navigation work on Actuator Networks, described in Chapter 7 has two distinct fundamental differences from previous approaches. Firstly, Actuator Networks do not require direct sensing of a robot's location after the first timestep, however the robot must always be able to sense the fields emitted from the actuators. Secondly, whereas past research has focused on modeling a robot's environment and navigation instructions with potential fields or other virtual forces that can take on arbitrary shape, our work considers a network of actuators where signals can only be generated at the actuator locations. As the actuators emit physical signals, rather than virtual ones, there is a limited class of fields which each actuator can emit. To overcome the additional restrictions of our problem, the potential fields in our Actuator Network formulation transition at discrete time steps, and problems related to local minima are avoided by steering the robot through waypoints, rather than directly from start to end location. As in SMR, we use the objective of maximizing probability of success over a roadmap. But unlike SMR, which assumes perfect sensing, the maximum probability path for an actuator network must be computed before plan execution is begun since sensing feedback is not available.

2.4 Probabilistic Modeling and Inference For Tracking

To localize and track the various objects for our applications, we need a theoretically sound framework to model these problems. In our applications, we have noisy sensor readings of the objects we track, and noisy information about how the objects move over time. We model both of these types of information using probability distributions. Our approaches are methods of inference, where we use observations, and noisy probabilistic models, to determine the posterior distributions over unobserved random variables. As guided by principles of Structured Tracking, we use dynamics models to improve the accuracy, robustness to difficult cases, and computational complexity of our estimates. We refer to the model of how a sensor responds to the stimulus of the object it observes as an observation model, and refer to the model of how the object moves over time as a dynamics model.

There are a number of methods for inference when the sensors directly provide a noisy estimate of the state of the object, and the object's new state is determined solely by the previous state. Distributions with this structure are Hidden Markov Models (HMM) [162]. One widely-used method for inference in continuous state HMMs, is Kalman Filtering [95], where distributions are limited to linear functions, with additive Gaussian noise. The Extended Kalman Filter (EKF) [115] and the Particle Filter [162] are approximate methods which can be applied to broader classes of models. The Extended Kalman Filter relaxes the linear and Gaussian constraints to distributions that are differentiable. It linearizes the distribution around the current estimate, and then uses the Kalman Filter's approach. In the work of this Dissertation, we focus on an alternative, non-parametric approach, Particle Filtering, where distributions are represented with a set of samples, and the density of samples is proportional to the probability density function (PDF) of the distribution it approx-

imates. The distinction between parametric and non-parametric approaches is that in parametric approaches, the parameters that describe a distributions do not scale with the amount of data. For instance, in a Gaussian distribution, it always only has two parameters, a mean and a standard deviation. In a non-parametric method, we represent a distribution with samples. In our applications, we frequently choose Particle Filtering because the distribution of the sensor's estimate of the tracked object's state is non-Gaussian, and thus methods like Kalman Filtering cannot be applied.

In some of our later localization and tracking work, we use different forms of Belief Propagation algorithms [148], as a means of inference. One such method, Nonparametric Belief Propagation (NBP) [183] can be viewed as a generalization of Particle Filters to domains with richer, non-temporal structure, or more precisely, graphical models that cannot be represented as an HMM. For example, we can use it to perform inference over spatial, probabilistic constraints between pairs of random variables. To make this more concrete, we can have random variables which represent object locations, and have noisy inter-distance sensor readings, which describe probabilistic constraints about how far neighboring sensors can be positioned relative to one another. We then can use these local constraints and Belief Propagation to determine the most likely configuration of objects, and thus, their locations.

2.4.1 Hidden Markov Model

A Markov Model has the property that the state of a system is only dependent on the previous state. More formally, the relations between all states have the property for timesteps $0, \ldots, \tau$:

$$P(X_\tau | X_{\tau-1}, \ldots, X_0) = P(X_\tau | X_{\tau-1}).$$

In a HMM, the state cannot be observed directly, and instead is performed indirectly through some correlated observation. For instance, sensory information could provide

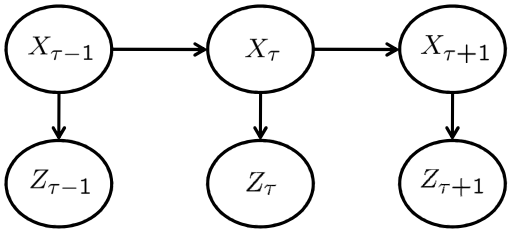


Figure 2.1. A graphical model of a Hidden Markov Model. The nodes in the graph represent random variables, and a directed edge represents dependence between random variables. Direct arrows denote which parent nodes directly influence child nodes, which provides a graphical representation of how to decompose a joint distribution. For more details, please see a summary on directed graphical models [162]. The arrows from all timesteps from state X to observations O indicate the noisy observation of the true state. The arrows for the state from one timestep to the next, for instance $X_{\tau-1}$ to the next X_τ, capture the Markov property.

a noisy estimate of an object's true state. The canonical example of a graphical model for an HMM is provided in Figure 2.1. More information about graphical models can be found in Section 2.4.4, where we focus on undirected graphical models, while Figure 2.1 uses a directed graphical model formulation.

2.4.2 Importance Sampling

Importance Sampling [49] is a fundamental technique used by both Particle Filtering and NBP approaches. For many inference tasks, we wish to estimate the target density function $p(X)$ using samples. However, for many of these distributions, we are unable to directly draw samples from $p(X)$. Fortunately, because we are often able to evaluate the contribution of a pdf at a specific state, which is often given in closed form, we are able to use the importance sampling approach to estimate the expectation of any function of the pdf $E(f(X))$. We draw samples $x^{(i)}$ from a proposal density function $q(X)$, which is chosen by those designing inference algorithms to be similar to the target density function. The more similar the functions, the fewer samples will be required to accurately estimate the expectation of the target density. As

determined by importance sampling methods, we use the proposal density function to assign a weight $w^{(i)} \propto p(x^{(i)})/q(x^{(i)})$ to each sample $x^{(i)}$. We can then estimate the expectation of $f(X)$ by weighting each state $x^{(i)}$ by $w^{(i)}$, which is referred to as the importance weight. This specific weighting allows us to draw samples to estimate the expectation of any function $f(x)$ because

$$\begin{aligned} E(f(X)) &= \int f(x)p(x)dx \\ &= \int f(x)q(x)\tfrac{p(x)}{q(x)}dx \\ &= \int f(x)q(x)w(x)dx \end{aligned} \quad .$$

Thus, these weighted samples describe all higher moments (or more generally significant statistics of any distribution). As the number of samples approaches infinity, the weighted samples approach the true distribution. In many instances, including Particle Filtering and NBP, these weighted samples are resampled with replacement, so that we have a number of samples with uniform weight, which represent the distribution, and the more samples there are in a specific state, the greater the PDF at that state.

2.4.3 Particle Filtering

While there are many types of Particle Filters, our approaches use the Sampling Importance Resampling (SIR) Filter as described in [12, 162]. It is a non-parametric method for performing state estimation of HMMs over discrete time. The state at the timestep τ is represented as a random variable x_τ and the evidence of the hidden state, in our case the sensor model, is represented as z_τ. There are three distributions needed for SIR Particle Filtering: the prior probability distribution of the object's state $P(x_0)$, the dynamics model (also referred to as a transition model) $P(x_\tau|x_{\tau-1})$, and the observation model (also referred to as a sensor model) $P(z_\tau|x_\tau)$. These describe the probability distributions for the edges in the graphical model illustrated

in Figure 2.1. The prior describes the initial distribution of the object's state. The transition model describes the distribution of the object's state at the next iteration, given the current object state. The observation model describes the distribution of observations resulting from a specific object's state. Particle Filtering uses a set of samples of the state, or "particles." At each iteration, each particle is advanced according to the transition model, and then assigned a probability according to its likelihood using the observation model. After all particles have a new likelihood, they are resampled with replacement using the relative probabilities determined via the observation model. This weighting and resampling makes use of importance sampling, where the proposal distribution, $q(x)$, is the observation model. The result of this approach is a distribution of new particles which have integrated all previous observations and are distributed according to their likelihood. After performing this form of inference, the samples are distributed proportionally with the posterior distribution $P(x_\tau|z_{0:\tau})$. Thus, the more samples that are within a specific state, the more likely that the indirectly observed object is actually in that state. A visual depiction of the algorithm is presented in Figure 2.2.

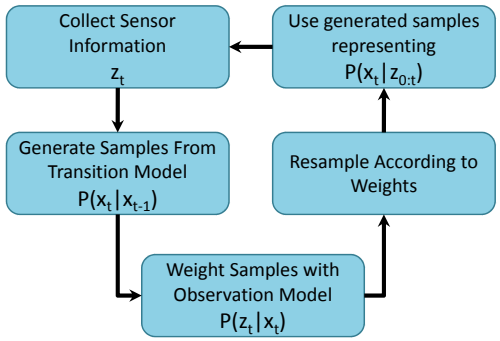


Figure 2.2. An overview of one iteration of Particle Filtering for tracking at time τ.

Derivation of Particle Filtering

Now that we have established importance sampling, we can illustrate how Particle Filtering is derived using importance sampling. In most Particle Filtering applications, it is possible to draw samples for the next timestep from an object's transition model, as the transition model describes what objects' new state will be given its previous state. We can achieve this because the object dynamics directly describes how the object might evolve over time. However, it is natural to describe an observation model in terms of a PDF. Therefore, we will use importance sampling where we transition the samples from the previous timestep to get our proposal distribution, and then weight these samples by the observation model to achieve the posterior distribution $P(x_\tau|z_{0:\tau})$ that we would like to estimate.

We have a set of samples drawn from $P(x_\tau|z_{1:\tau})$ from the previous iteration. We transition these samples according to the transition model $P(x_{\tau+1}|x_\tau)$, which results in a new distribution of samples distributed according to $P(x_{\tau+1}|z_{1:\tau})$. We then weight each newly transitioned sample by the observation model $P(z_{\tau+1}|x_{\tau+1})$. According to importance sampling, because the weight $w^{(i)} = p(x^{(i)})/q(x^{(i)})$, we know that the target distribution $p(x^{(i)}) = w^{(i)}q(x^{(i)})$. By weighting and resampling the transition model, the samples are distributed proportionally with $P(z_{\tau+1}|x_{\tau+1})P(x_{\tau+1}|z_{1:\tau})$ which is also equal to $P(x_{\tau+1}|z_{1:\tau+1})$, as shown in Proposition 1. The α term in Proposition 1 is a normalizing constant, as $P(z_{\tau+1}|x_{\tau+1})P(x_{\tau+1}|z_{1:\tau})$ does not describe a probability without ensuring that the sum of the probabilities of all states equals 1. This is accomplished implicitly in the reweighting step, as we make sure that the sum of the weights of all samples equals 1. Thus, the weighting and resampling step gives us the distribution we wish to estimate $P(x_{\tau+1}|z_{1:\tau+1})$.

Proposition 1.

$$\alpha P(z_{\tau+1}|x_{\tau+1})P(x_{\tau+1}|z_{1:\tau}) = P(x_{\tau+1}|z_{1:\tau+1}),$$

Proof.

$$
\begin{aligned}
& P(x_{\tau+1}|z_{1:\tau+1}) \\
& = P(x_{\tau+1}|z_{1:\tau}, z_{\tau+1}) \\
& \text{(using Bayes rule)} \\
& = \tfrac{P(x_{\tau+1}, z_{1:\tau}, z_{\tau+1})}{P(z_{1:\tau}, z_{\tau+1})} \\
& = \tfrac{P(z_{\tau+1}|x_{\tau+1}, z_{1:\tau})P(x_{\tau+1}|z_{1:\tau})P(z_{1:\tau})}{P(z_{\tau+1}|z_{1:\tau})P(z_{1:\tau})} \\
& = \tfrac{P(z_{\tau+1}|x_{\tau+1}, z_{1:\tau})P(x_{\tau+1}|z_{1:\tau})}{P(z_{\tau+1}|z_{1:\tau})} \\
& = \alpha P(z_{\tau+1}|x_{\tau+1}, z_{1:\tau})P(x_{\tau+1}|z_{1:\tau}) \\
& \text{(by the Markov Property of the Evidence Model)} \\
& = \alpha P(z_{\tau+1}|x_{\tau+1})P(x_{\tau+1}|z_{1:\tau})
\end{aligned}
\tag{2.1}
$$

■

2.4.4 Inference over Undirected Graphical Models

Graphical modeling theory allows us to describe problems that are more general than HMMs. As shown in Figure 2.1, HMMs can be described by a specific graphical model. Inference techniques over graphical models allow us to address a broader class of problems. We begin by providing background on graphical models, and then turn to discuss message-passing algorithms, including the standard and reweighted sum- and max-product algorithms, and a non-parametric sum-product algorithm.

Undirected Graphical Models

We model the relationships among the random variables in some of our tracking problems using an undirected graphical model, or pairwise Markov random field (MRF) [93]. This is in contrast to directed models, such as those used to illustrate

an HMM in Figure 2.1. This model is specified via an undirected graph G, with vertex set $V = \{1, \ldots, n\}$ and edge set E. Each node $s \in V$ is associated with a random variable x_s representing (for our purposes) some type of sensor measurement or a quantity of interest, such as the temperature in a particular location or a pallet's location. These random variables can take values in a discrete space (e.g., $\mathcal{X} = \{0, 1, \ldots, m-1\}$), or in some continuous space (e.g., $\mathcal{X} = \mathbb{R}$). The distributions of discrete random variables are typically represented by histograms, and distributions of continuous random variables are often represented either with a small number of parameters (like a Gaussian), or using a non-parametric approach (such as samples). For notational simplicity, when we discuss algorithms that involve only a single timestep, like in Section 5.3 on static localization, we denote node variables as x_s. For the tracking problems considered in Section 5.4, they become temporal states $x_{s,\tau}$.

The graph G specifies a factorization of the joint distribution of these random variables into a product of local, non-negative compatibility functions. In particular, for the collection of random variables $\vec{X} = (X_1, \ldots, X_n)$, we associate with each vertex s a function $\psi_s : \mathcal{X} \to \mathbb{R}_+$ (called single-site compatibility function), and with each edge (s, t) a function $\psi_{st} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ (called edgewise compatibility function). Under the MRF assumption, the joint distribution of $\vec{X}$ factorizes as

$$p(\vec{x}) = \frac{1}{Z} \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t) \tag{2.2}$$

Here, Z is a normalization constant, and the compatibility functions encode the information provided by dynamics models and observed sensor readings. Although it can be convenient to also define compatibility functions on larger subsets $|S| > 2$ of nodes, there is no loss of generality in making the pairwise assumption (2.2).

Statistical inference in MRFs

Given an MRF of the form (2.2), it is frequently of interest to compute marginal probabilities at a particular node (or over some subset of nodes). Such marginalization problem involves summing over (subsets of) configurations, to determine marginal probability distributions. In our problem, this would result in the marginal distributions of the locations of each robot at each timestep given the observed inter-distance readings. This problem is computationally challenging because the number of terms in the summation grows exponentially in the size of the network (more specifically, $|\mathcal{X}|^n$ for a discrete MRF). For graphs with special structure—in particular, trees and more generally graphs of bounded treewidth[1]—the marginalization problem can be solved exactly with the junction tree algorithm [110], albeit with exponential complexity in treewidth. Paskin et al. [147] developed and implemented a robust architecture for the junction tree algorithm, suitable for performing exact inference in sensor networks. However, many connectivity graphs topologies commonly used to model sensor networks, including grid-based graphs and random geometric graphs, have unbounded treewidth, so the corresponding graphical models have cycles. Because the junction tree inference algorithm is exponential in treewidth, approximate algorithms are needed in practice.

Belief Propagation Algorithms

Belief Propagation algorithms (BP) [148] are a general class of inference algorithms for graphical models. BP algorithms perform inference via message passing, where each node s in the MRF iteratively integrates information via a local computation, and then transmits a summary message to each neighbor $t \in \Gamma(s)$, hence, the belief propagates from random variable to random variable. In general, this message

[1]In loose terms, a graph of bounded treewidth is one which consists of a tree over clusters of nodes; see the paper [110] for further details.

$M_{st}(x_t)$ encodes the sufficient statistics from node s needed for node t to perform the next round of computation. There are different variants of Belief Propagation algorithms for inference, including the *sum-product* updates for approximate marginalization, and the *max-product* updates for approximate maximization (or MAP). For tree-structured graphs (and with modifications for junction trees), both forms of these message-passing algorithms are dynamic programming algorithms, which exactly compute the posterior marginal distributions $p_s(x_s)$ for all nodes $s \in V$. The same updates are widely applied to general graphs with cycles, in which case they provide approximate solutions to inference problems. When executed on graphs with cycles as approximation algorithms, they are often called "loopy" Belief Propagation. In our approaches, our formulations and experiments use the sum-product algorithm, but other Belief Propagation approaches like max-product could just as easily be applied.

In 1988, loopy BP was suggested by Pearl[148] (see [138] for historical discussion) as part of the artificial intelligence community. However, there was limited adoption of the technique, due to concerns that it was just an approximation algorithm, and no formal guarantees could be provided about the algorithm's convergence or accuracy on graphs with cycles. In 1993, turbo codes were independently discovered [18], and provided decoding performance near the Shannon limit. This was a dramatic advancement, as previous codes were unable to achieve similar performance. In the following years, researchers showed the equivalence between turbo codes and loopy BP applied to a specific graphical model [67, 127]. The large improvement of turbo codes over previous algorithms has led to the application of loopy BP in a broad range of other machine learning and statistical inference problems [40, 64, 66, 138, 157, 185, 218]. Due to its distributed nature, BP has also been applied to many sensor network applications [88, 34, 41, 137, 120]. The recent survey paper by Cetin et al. [31] (and references therein) provides further discussion of the issues that arise with BP in sen-

sor networks. BP's general framework and effectiveness at solving inference problems has spurred research into understanding why the algorithm works so effectively, and ways of augmenting the algorithm to get better convergence and accuracy guarantees, which we summarize in the Reweighted Belief Propagation subsection of Section 2.4.4.

The BP algorithm begins by initializing all messages $M_{st}(x_t)$ to constant vectors, and then updates the messages along each edge according to the following recursion:

$$M_{st}(x_t) \leftarrow \int_{x_s} \psi_s(x_s)\psi_{st}(x_s, x_t) \prod_{u \in N(s)\setminus t} M_{us}(x_s)\, dx_s \tag{2.3}$$

For discrete variables x_s, messages are represented by finite vectors, and the integral in Equation (2.3) becomes a summation. This BP algorithm is then iterated until the set of messages converges to some fixed point. The order in which the messages are updated is a design parameter, and various schedules exist. These schedules can affect the speed of convergence and can possibly result in different fixed points. Upon convergence, the algorithms can be used to compute approximations to the marginal distributions at each node:

$$\hat{p}_s(x_s) \propto \psi_s(x_s) \prod_{t \in N(s)} M_{ts}(x_s) \tag{2.4}$$

Nonparametric BP: Distributions as Samples

The computation of the Belief Propagation is straight-forward for formulations using discrete or Gaussian potential functions. However, in domains such as localization and tracking with only inter-distance estimates, Gaussian models are too restrictive to correctly model the distributions. Discrete distributions have issues with scalability, and we compare discrete BP versus NBP in more detail in Section 5.1. Given the limitations of previous approaches, we employ Nonparametric Belief Propagation (NBP) [183], which directly approximates the marginal distributions of continuous states via a collection of M sample points. Because we use importance sampling, as

described in Section 2.4.2, to incorporate the information from incoming messages to compute marginals, we need a way of determining a PDF from incoming message samples. We accomplish this by smoothing each sample, turning each into a Gaussian. Thus, messages M_{st} from node s to node t in our graphical model are represented via a weighted mixture of M Gaussian distributions

$$M_{st}(x_t) = \sum_{i=1}^{M} w_{st}^{(i)} \mathbb{N}(x_t \mid x_{st}^{(i)}, \Sigma_{st}) \tag{2.5}$$

Mixture components are centered on samples $x_{st}^{(i)}$ from the underlying, continuous message function, with weights $w_{st}^{(i)}$ set via importance sampling principles as detailed below. By choosing a number of samples M, we can tradeoff accuracy versus computational cost.

A variety of methods are available for choosing the covariance Σ_{st} used to smooth message samples [175] for importance sampling. In many cases, we use the computationally efficient "rule of thumb" estimate $\Sigma_{st} = \text{ROT}(\{x_{st}^{(i)}, w_{st}^{(i)}\})$, which is proportional to the weighted covariance of the observed samples:

$$\Sigma_{st} = M^{\frac{-2}{\delta+4}} \sum_{i=1}^{M} w_{st}^{(i)} (x_{st}^{(i)} - \bar{x}_{st})(x_{st}^{(i)} - \bar{x}_{st})^T \tag{2.6}$$

Here, $\bar{x}_{st} = \sum_i w_{st}^{(i)} x_{st}^{(i)}$ is the weighted sample mean, and $\delta = \dim(x_t)$. However, for ring-shaped messages as occur in inter-distance sensor distributions, which are far from unimodal, the rule of thumb estimator performs poorly, significantly oversmoothing the estimated density. In such cases, we hardcode Σ_{st} as a function of our noise and our number of samples. For instance in our Sparse Perceptive Pallets problem, we set $\Sigma_{st} = \sigma_\nu^2 \xi_M I$, where σ_ν^2 is the variance of the noise added to inter-distance sensor readings of neighbors. ξ_M is a constant calibrated offline to the number of samples M.

Reweighted Belief Propagation

In this section, we first describe the family of *Reweighted Belief Propagation* (RBP) algorithms, and then illustrate how a particular choice of edge weights yields the standard Belief Propagation algorithm. Various researchers have studied and used such reweighted algorithms for the sum-product updates [117, 202], generalized sum-product [208], and max-product updates [203, 104, 129, 217]. RBP is a larger family of algorithms, which includes as special cases the standard BP and max-product algorithms for general networks with cycles. Each algorithm in this family is specified by a vector $\vec{\rho} = \{\rho, (s,t) \in E\}$ of edge weights. The choice $\rho_{st} = 1$ for all edges $(s,t) \in E$ corresponds to standard Belief Propagation; different choices of $\vec{\rho}$ yield distinct algorithms with convergence, uniqueness, and correctness and robustness to message error guarantees for RBP [202, 203] and correctness as well as convergence guarantees for reweighted max-product [104, 217, 203]. These properties are *not* shared by the standard BP algorithm for general networks. Indeed, as illustrated in Figure 3.1, for certain network structures, the standard BP algorithm can yield highly inaccurate and unstable solutions to inference problems. In contrast to this instability, appropriately designed RBP algorithms are theoretically guaranteed to be robust to both message errors, and model mis-specification. For any fixed set of edge weights $\vec{\rho}$, the associated reweighted BP algorithm begins by initializing all of the messages M_{st} to constant vectors; the algorithm then operates by updating the message along each edge according to the recursion

$$M_{st}(x_t) \leftarrow \int_{x_s} \psi_s(x_s) \left[\psi_{st}(x_s, x_t)\right]^{\frac{1}{\rho_{st}}} \frac{\prod_{u \in N(s)\backslash t} [M_{us}(x_s)]^{\rho_{us}}}{[M_{ts}(x_s)]^{1-\rho_{st}}}. \tag{2.7}$$

These updates are repeated until the vector of messages $\vec{M} = \{M_{st}, M_{ts} \mid (s,t) \in E\}$ converge to some fixed vector $\vec{M}^*$. The order in which the messages are updated is a design parameter, and various schedules (e.g., parallel, tree-based updates, etc.) exist. These schedules affect the speed of convergence and can possibly result in

different fixed points. Upon convergence, the message fixed point $\vec{M}^*$ can be used to compute approximations to the marginal distributions at each node and edge via

$$q_s(x_s) \quad \propto \quad \psi_s(x_s) \prod_{t \in N(s)} [M_{ts}(x_s)]^{\rho_{st}}, \tag{2.8}$$

and also to generate an approximation to the normalization constant. The update (2.7) corresponds to the sum-product algorithm; replacing the summation $\sum_{x_s}$ by the maximization $\max_{x_s}$ yields the reweighted form of the max-product algorithm.

For the special setting of unity edge weights $\rho = 1$, the update equation (2.7) corresponds to the standard Belief Propagation algorithm (2.3). For any tree-structured graph, this algorithm is exact, and can be derived as a parallel form of dynamic programming on trees. On graphs with cycles, neither standard Belief Propagation (nor its reweighted variants) are exact. In this context, one interpretation of Belief Propagation is as a Lagrangian method for solving the so-called Bethe variational problem (see [219] for more details). This interpretation has proven to be a useful method for understanding properties about Belief Propagation such as convergence, and understanding why it works so frequently in practice.

Algorithmic stability and robustness

In the case of standard BP ($\rho_{st} = 1$ for all edges (s,t)), there can be many fixed point solutions for the updates (2.7), and the final solution can depend heavily on the initialization. In contrast, for suitable settings of the weights [202] different from $\rho_{st} = 1$, the updates are guaranteed to have a unique fixed point *for any* network topology and choice of compatibility functions. Moreover, such reweighted BP algorithms are known to be *globally Lipschitz stable* in the following sense [202]. Suppose that $q(\psi)$ denotes the approximate marginals when a RBP algorithm is used for approximate inference with data (compatibility functions) ψ. Then there is a global constant L, depending only on the MRF topology, such that $\|q(\psi) - q(\psi')\| \leq L\|\psi - \psi'\|$, where

$\|\cdot\|$ denotes any norm. In loose terms, this condition guarantees that bounded changes to input yield bounded output changes, which is clearly desirable when applying an algorithm to statistical data. For special choices of compatibility functions (but not all choices), the standard BP updates are also known to be stable in this sense [188, 86]. Our experimental results described in Section 3.4 confirm the desirability of such stability.

Chapter 3

Safety: StatSense

3.1 Introduction

Recent advances in hardware and software are leading to sensor network applications with increasingly large numbers of motes. In such large-scale deployments, the straightforward approach of routing all data to a common base station may no longer be feasible. Moreover, such an aggregation strategy—even when feasible—can be wasteful, since it ignores which aspects of the data are relevant (or irrelevant) to addressing a given query. Consequently, an important research challenge is the development and practical implementation of distributed algorithms for performing data fusion in an in-network manner, thereby leading to useful statistical summaries of sensed measurements [222, 31].

Various problems that arise in sensor network applications, ranging from estimation and regression (e.g., predicting a "smoothed" version of a temperature gradient) to hypothesis testing (e.g., determining whether or not a fire has occurred), are particular instances of statistical inference. Past work on sensor networks [34, 31, 41, 120, 137, 147] has established the utility of formulating such in-

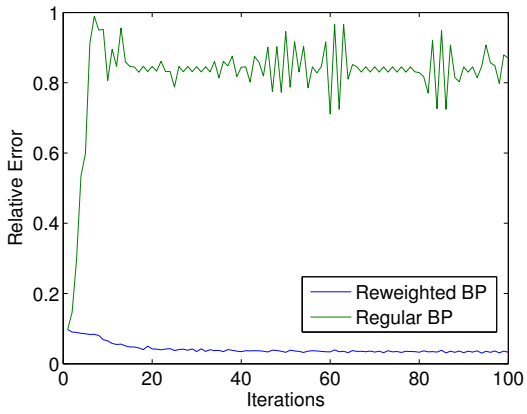


Figure 3.1. Graph of error versus iteration number for message-passing on a 9×9 nearest-neighbor grid. Standard Belief Propagation (BP, green curve with high error) vs. reweighted Belief Propagation (RBP, blue curve with low error).

ference problems in terms of Markov random fields, a type of graphical model in which vertices represent variables of interest and edges correspond to probabilistic constraints between them. Moreover, given such a graphical model, it is possible to define simple message-passing algorithms for performing inference, specifically variants of Belief Propagation, in which any given node passes "messages" to its neighbors that represent statistical summaries of local information relevant to a global computation. In a sensor network scenario, each mote is assigned a subset of variables of the Markov random field and we use the distributed wireless network of motes to execute the message-passing algorithm. The fact that Belief Propagation algorithms for graphical models require no global coordination translates to very simple and robust message passing algorithms for the sensor motes. The mapping of the graphical model to sensor motes and related issues are discussed in subsequent sections.

For statistical inference, a number of researchers [41, 137, 120] have studied the use of standard BP for sensor networks at both the theoretical and simulation level. However, the work described here is (to the best of our knowledge) the first to actually implement a class of loopy message-passing algorithms—including BP as one exemplar—for sensor motes. More specifically, we design and implement an architec-

ture for implementing RBP algorithms in real sensor networks. Our design does not rely on infrastructure such as reliable messaging, time synchronization and routing. We present simulation results evaluating algorithm performance under the real-world problems of failing motes and communication problems of failing links, asymmetric links, and dropped messages. We also show that intelligent scheduling, with greater communication between nodes on the same mote than nodes across motes, can make the algorithms converge with much less communication. We present experimental results from a prototype implementation using Mica2 motes where we infer the temperature at unobserved locations within 3 degrees, over a 45 degree temperature range. Our nesC implementation is modular and can be easily adapted for any message passing algorithm and scheduling policy, which as we show, is useful for numerous applications.

3.2 Proposed Architecture

In the following section, we present *StatSense*, an architecture for implementing RBP algorithms in sensor networks. Our primary focus is the challenges that arise in implementing message-passing algorithms over unreliable networks with severe communication and coordination constraints.

3.2.1 Mapping from graphical models to motes

As described in Section 2.4.4, any Markov random field (MRF) consists of a collection of nodes, each representing some type of random variable, joined by edges that represent statistical correlations among these random variables. Any sensor network can also be associated with a type of communication graph, which in general may be *different* than the MRF graph, in which each vertex represents a mote and

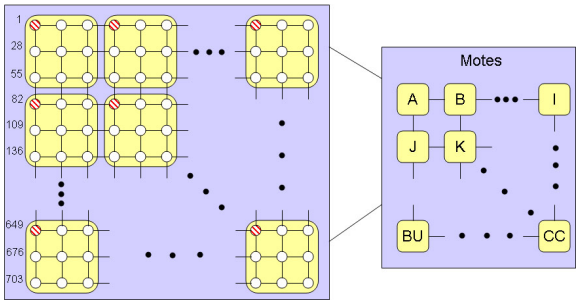


Figure 3.2. Illustration of the two layers of the graphical model (left-hand subplot) and motes (right-hand subplot) used for simulation. Each node in the graphical model is mapped to exactly one mote, whereas each mote contains some number (typically more than one) of nodes. The red angle-slashed dots denote observation nodes (i.e., locations with sensor readings).

the links between motes represent communication links. As we discuss here, there are various issues associated with the mapping between the MRF graph and the sensor network graph [31].

For sensor network applications, some MRF random variables will be associated with measurements, whereas other variables might be *hidden* random variables (e.g., temperature at a room location without any sensor, or an indicator variables for the event "Room is on fire"). Any observable node may be associated with multiple measurements, which can be summarized in the form of a histogram representing a probability distribution over the data. In this context, the role of message-passing is to incorporate this partial information in order to make inferences about hidden variables.

We assume that each node s in the MRF is mapped to a unique mote $\Gamma(s)$ in the sensor network; conversely, each mote A is assigned some subset $\Lambda(A)$ of MRF nodes. For any node $s \in \Lambda(A)$, the mote A is responsible for handling all of its computation and communication with other nodes t in the MRF neighborhood $N(s)$. Each node $t \in N(s)$ might be mapped either to a different mote (requiring communication across motes) or to the same mote (requiring no communcation). Figure 3.2 illustrates one example assignment for a sensor network with 81 motes, and an MRF with 729 nodes.

For instance, in this example, mote A is assigned nodes 1,2,3,26,27,28, 55, 56 and 57, so that $\Lambda(A) = \{1, 2, 3, 26, 27, 28, 55, 56, 57\}$, and $\Gamma(1) = A$ etc. The sensor associated with each mote corresponds to an evidence node in the MRF; for instance, {1, 4, ..., 82, ..., 673} are evidence nodes in this example and we represent them as hashed nodes in Figure 3.2. Similar to previous work [128, 147], we assume a semi-static topology in creating the mote communication graph, so that we place an edge in this graph if there is a high-quality wireless communication link between these two motes.

Note that the assignment of MRF nodes to motes may have a substantial impact on communication costs, since only the messages between nodes assigned to different motes need be transmitted. There is an additional issue associated with the node-mote assignment: in particular, the following property is necessary to preserve the distributed nature of message-passing when implemented on the sensor network link graph: for any pair (s, t) of nodes joined by the edge in the MRF, we require that the associated motes $\Gamma(s)$ and $\Gamma(t)$ are either the same ($\Gamma(s) = \Gamma(t)$), or are joined by an edge in the mote communication graph. We refer to this as the *no-routing* property, since it guarantees that message-passing on the MRF can be implemented in motes with only nearest-neighbor mote communication, and hence no routing. Thus, for a given MRF and mote communication graph, a question of interest is whether the no-routing property holds, or can be made to hold. It is straightforward to construct MRFs and mote graphs for which it fails. However, the following result guarantees that it is always possible to modify the MRF so that this property holds:

Proposition 2. *Given any MRF model and connected mote communication graph, it is always possible to define an extended MRF, which when mapped onto the same mote communication graph, satisfies the no-routing property, and that message-passing on the mote communication graph yields equivalent inference solutions to the original problem.*

Proof. Our proof is constructive in nature, based on a sequence of additions of nodes to the MRF model such that: *(a)* the final extended model satisfies the no-routing property, and *(b)* running message-passing on the mote communication graph yields identical inferences to message-passing *with routing* in the original model. Throughout the proof, the set of motes $A, B, C, \ldots$ and the associated mote communication graph remains fixed. The number of nodes and compatibility functions in the MRF as well as the mapping from nodes to motes are quantities that vary. Given any MRF model with variables $(X_1, \ldots, X_n)$ and mote assignments $(\Gamma(1), \ldots, \Gamma(n))$, suppose that the no-routing property fails for some pair (s, t), meaning that pair of motes $\Gamma(s)$ and $\Gamma(t)$ are distinct, and *not* joined directly by an edge in the mote link graph. Since the mote communication graph is connected, we can find some path $\mathcal{P} = \{\Gamma(s), A_2, \ldots, A_{p-1}, \Gamma(t)\}$ in the mote communication graph that joins $\Gamma(s)$ and $\Gamma(t)$. Now for each mote $A_i,\ i = 2, \ldots (p-1)$, we add a new random variable Y_i to the original MRF; each random variable Y_i is mapped to mote A_i (i.e., $\Gamma(Y_i) = A_i$)). Moreover, let us remove the compatibility function $\psi_{st}(x_s, x_t)$ from the MRF factorization (2.2), and add to it the following compatibility functions

$$\begin{aligned}
\widetilde{\psi}_{s\,2}(x_s, y_2) &= \mathbb{I}\,[x_s = y_2] \\
\widetilde{\psi}_{k\,(k+1)}(y_k, y_{k+1}) &= \mathbb{I}\,[y_k = y_{k+1}]\,, \qquad \text{for } k = 2, \ldots, (p-2). \\
\widetilde{\psi}_{(p-1)\,t}(y_{p-1}, x_t) &= \psi_{st}(y_{p-1}, x_t).
\end{aligned}$$

Here the function $\mathbb{I}(a, b)$ is an indicator for the event that $\{a = b\}$. The basic idea is that the variables $(Y_2, \ldots, Y_{p-2})$ represent duplicated copies of X_s that are used to set up a communication route between node s and t. By construction, the communication associated with each of the new compatibility functions $\widetilde{\psi}$ can be carried out in the mote graph without routing. Moreover, the new MRF has no factor $\psi_{st}(x_s, x_t)$ that directly couples X_s to X_t, so that edge (s, t) no longer violates the no-routing property. To complete the proof, we need to verify that when message-

passing is applied in the mote graph associated with the new MRF, we obtain the same inferences upon convergence (for the relevant variables $(X_1, \ldots, X_n)$) as the original model. This can be established by noting that the indicator functions $\mathbb{I}$ collapse under summation/maximization operations, so that the set of message-fixed points for the new model are equivalent to those of the original model.

Thus, we have shown that an edge (s, t) which fails the no-routing property can be disposed of, without introducing any additional links. The proof is completed by applying this procedure recursively to each troublesome MRF pair (s, t). ■

Note that while one can design MRF models and corresponding communication graphs that contain a quadratic number of problematic edges, in practically interesting models, this transformation will only be rarely required, as discussed in Section 3.5.

3.2.2 Message updating

As mentioned previously, the message update scheme is a design parameter. The most common message update scheme is one in which all nodes update and communicate their messages at every time step according to (2.7), and proceed synchronously to new iterations. We term this scheme *SyncAllTalk*.

Unfortunately, several factors discourage the direct application of SyncAllTalk to sensor networks. First, radio transmission and reception on embedded hardware devices consume nontrivial amounts of energy. Passing messages *inter-mote* is expensive and may overburden the already resource-constrained network. On the other hand, passing messages *intra-mote* incurs no significant energy cost because it is entirely local. This dichotomy indicates that we should limit messaging across motes when possible. Second, the SyncAllTalk protocol relies on synchronous message passing

that inherently exhibits "bursty" communication patterns. For shared communication channels such as wireless, burstiness results in issues such as the *hidden terminal problem*, which further exacerbates the cost of radio transmission and reception. Last, for many situations, we expect sensor informativeness to vary greatly. For example, in a building monitoring scenario, the first sensor to detect a fire will generate much more informative messages than other sensors. Other scenarios have also resulted in similar observations [147]. Thus, evidence nodes, which often correspond directly to sensors, vary greatly in informativeness. We would like to favor more informative messages and thereby accelerate the rate of convergence.

We have investigated a number of schemes which take advantage of these factors. The first scheme, *SyncConstProb*, exchanges all intra-mote messages at each time step as before, but only exchanges each inter-mote message with probability p at each time step. This means there is a direct decrease in the average period of inter-mote message transmission from once every time step to once every $1/p$ time steps. This scheme can trade convergence time for communication overhead. Probabilistic sending also reduces the burstiness that causes the hidden terminal problem.

We can further decrease burstiness by relaxing the global time constraint. Instead, each mote proceeds at its own local start time and clock rate. As an additional benefit, this scheme does not rely on any time synchronization service. We refer to this scheme as *AsyncConstProb*.

Our last scheme, *AsyncSmartProb*, makes the message transmission probability proportional to the informativenss of the message. Rather than reference a global constant probability of sending, p, each host sends with probability determined by some measure of distance between the old and new messages (cf. [31] for related ideas). One possible metric that we explore here is *total variation distance* raised to

power η between the old and new messages:

$$p_{send}(M'_{st}) = \left(\frac{1}{2} \sum_{x_t} |M'_{st}(x_t) - M_{st}(x_t)| \right)^{\eta} \in [0, 1]. \tag{3.1}$$

Here M'_{st} denotes a new message that we may want to send from node s to node t, M_{st} denotes the previously sent message from node s to node t and η is a tunable "politeness" factor. Note that if $\eta \approx 0$, then $p \approx 1$ so that messages are almost always sent, whereas for very large η, only the most informative messages have a substantial probability of being transmitted.

All of the message passing schemes we investigate are extremely simple to implement, requiring no additional service infrastructure and operating exclusively with local computation. The traditional BP literature has proposed more sophisticated schemes in which all nodes exchange messages along graph overlays in organized sequence e.g. from the root of the overlay breadth-first. Unfortunately, these schemes typically require global coordination and are thus not as readily applicable in a sensor network context.

3.2.3 Handling communication failure

In general, when adapting algorithms to run on sensor networks, one must deal with network problems such as unreliable or asymmetric network links and mote failure. In the case of RBP, however, little adaptation is required. When a mote fails, the system simply ends up solving a *perturbed* inference problem on a reduced graphical model with all the nodes that belonged to that mote removed. If a network link fails, the resulting computation is equivalent to removing the corresponding edges from the graphical model. For both the case of mote and edge removals, we find empirically that the resulting increase in inference error remains localized near the removed nodes (or edges). The boundedness and localization of the error is again

consistent with theoretical results on the Lipschitz stability of RBP algorithms [202, 159] (see also the Algorithmic Stability and Robustness subsection of Section 2.4.4), in that perturbations cause bounded changes in the algorithm output. We also see in our evaluation (see Section 3.4) that causing links to be asymmetric induces limited error.

3.3 Implementation

In order to provide a robust and easily extensible system for performing RBP on sensornets, we implemented a general Belief Propagation framework in TinyOS/nesC. A diagram of our system is shown in Figure 3.3.

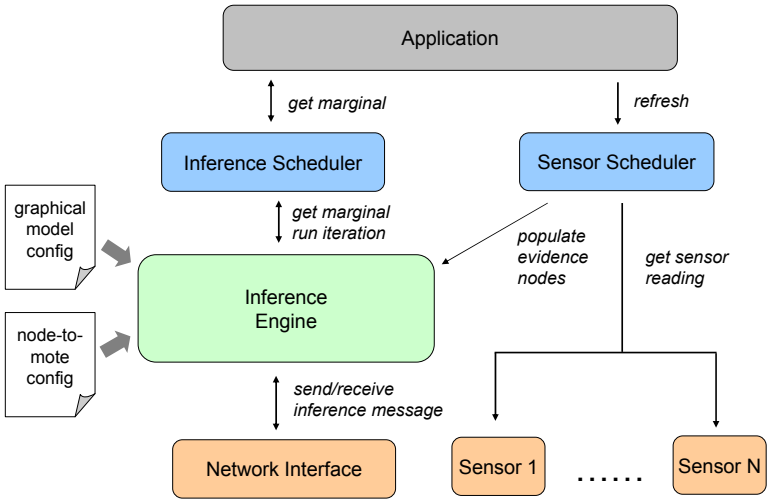


Figure 3.3. The StatSense architecture as implemented in nesC.

The core component of our design is the inference module. It receives incoming messages from other motes via the network interface module, new sensor readings via the sensor scheduler module, and instructions on when to compute new messages from the inference scheduler module. When instructed to, it uses the incoming messages and sensor data to compute its outgoing messages and sends any inter-mote

message to their destination via the network interface module. It handles intra-mote communication via loopback.

The network module provides very simple networks services such as buffering, marshalling and unmarshalling for incoming and outgoing messages. As a result of the robustness of RBP, we do not need any reliabile delivery. It is also the natural place to extend to more advanced network services such as data-centric multi-hop routing for cases of sophisticated node-to-mote mappings.

The sensor scheduler determines how often sensor readings are taken. Likewise, the inference scheduler determines when the inference module computes new messages according to one of the scheduling scheme detailed in Section 3.2.2. The inference scheduler also provides marginals of the graphical model, wrapping the functionality of the inference module.

We permit the user to input graphical models via configuration files. We created a pre-processor that converts this file into a nesC header file that contains all the data that the inference module needs to perform inference such as the compatability functions and node to mote mappings. Thus, it is straightforward for any StatSense user to build new graphical models and to use the outputs (marginals) of this model in her application.

We also architected the system in a modular fashion such that users are free to design new functional pieces independently. For example, the inference module, which currently supports BP, is easy to swap out for different message passing algorithms such as Max-Product, Min-Sum, *etc.* This allows those with a backgrounds in graphical modeling to test new algorithms without familiarity with sensornet systems issues. Likewise, it is easy to explore different scheduling schemes by replacing the inference scheduler and sensor scheduler.

3.4 Evaluation

In this section, we evaluate the performance of StatSense and RBP, the main message-passing algorithm investigated in the StatSense framework. We are primarily interested in understanding: (1) the impact of sensor noise on inference results; (2) the resilience of inference in the face of changing network conditions; and (3) inference convergence under different scheduling schemes.

We used both simulation and a Mica2 mote testbed for our experimental platform. We define error to be the average over all nodes of the difference between the ground-truth reading and the mean of the distribution described by the corresponding random variable. In all places where appropriate, we use a confidence interval of 95%. For simulation, we determine that our system has converged when the average L1 distances between the old and the new outgoing message over all edges in the graphical model drops below a threshold. In our experiements, we used the threshold of 0.005. For our deployment, we ran inference for 30 iterations.

As a summary of our evaluation, we highlight:

(a) StatSense with RBP shows resiliency to many types of network failures. As failures increase, error grows linearly with no sharp increases, and maintains a low absolute value. This empirical behavior is consistent with the theoretical stability guarantees discussed in the Algorithmic Stability and Robustness subsection of Section 2.4.4 .

(b) Improved scheduling schemes offer substantial, and in some cases, up to 50% fewer messages over naive scheduling schemes.

(c) Online inference in our testbed deployments exhibit accurate estimates of the ground truth.

3.4.1 The Temperature Estimation Problem

In order to evaluate our system, we examine RBP's application to temperature monitoring. In simulation, we model the room as a 27x27 node grid, similar to Figure 3.2. For each of the 100 training and 10 test runs, we randomly choose a "hot" and "cold" source placed as a 2x2 cell on this grid. We use the standard discretized heat diffusion process to calculate realistic steady-state temperatures which act as ground-truth. Since temperature has strong spatial correlations, our graphical model is a lattice, where each of the discrete locations in the room has a corresponding node in the graphical model. The edges in the model connect each node to its four nearest neighbors in a grid pattern. We discretize the temperature which ranges from 0 to 100 degrees into an eight bucket histogram.

Denote the ith observed temperature reading at location s to be $y_s^{(i)}$. Then, given M i.i.d. samples $y^{(i)} = \{y_1^{(i)}, \ldots, y_M^{(i)}\}$, we determine the compatibility functions empirically according to the following procedure. We first compute the empirical marginal distributions

$$\bar{\mu}_{st}(x_s, x_t) = \frac{1}{M} \sum_{i=1}^{M} \delta(x_s = y_s^{(i)}) \delta(x_t = y_t^{(i)}). \tag{3.2}$$

We then estimate the compatibility functions via the equations

$$\widehat{\psi}_s(x_s) = \frac{1}{M} \sum_{i=1}^{M} \delta(x_s = y_s^{(i)}) \tag{3.3}$$

$$\widehat{\psi}_{st}(x_s, x_t) = \left(\frac{\bar{\mu}_{st}(x_s, x_t)}{\widehat{\psi}_s(x_s) \widehat{\psi}_t(x_t)} \right)^{\rho_{st}} \tag{3.4}$$

These compatibility functions correspond to the closed-form maximum-likelihood estimate for tree-structured graphs, and have an interpretation as a pseudo-maximum-likelihood estimate for general graphs [202]. For both simulated and deployed experiments, we used the weights ρ_{st} for the torus topology which very closely approximate the weights for a grid.

We arrange our motes in a 9x9 grid-pattern, and perform a straightforward mapping of each 3x3 subgraph of the graphical model onto the corresponding motes. Each mote is physically located at the top left of its subgraph and is responsible for computation of messages for the other eight nodes assigned to it (as in Figure 3.2).

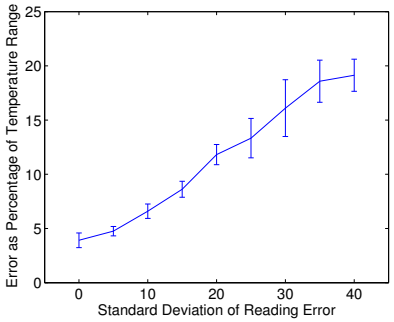


Figure 3.4. Average error as the Gaussian error applied to the sensor observations increases.

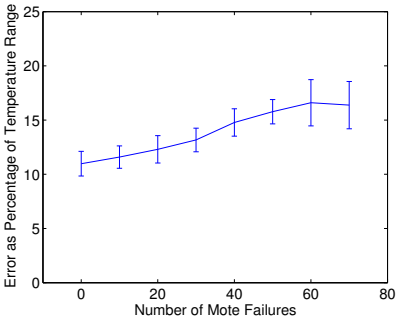


Figure 3.5. Average error as the number of dead motes increases.

3.4.2 Resilience to Sensor Error

We begin our tests by evaluating how sensor reading error affects inference performance. We induce unbiased Gaussian error to each temperature reading, with increasing standard deviation. We observe in Figure 3.4 that the algorithm performs well as error increases linearly, exhibiting no sharp threshold of breakdown. This is

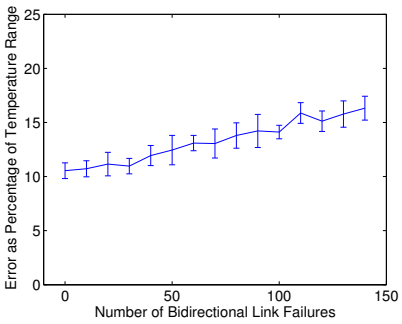


Figure 3.6. Average error as the number of dead symmetric links increases.

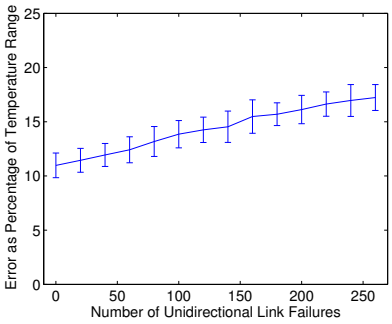


Figure 3.7. Average error as the number of dead asymmetric links increases.

because RBP fuses data using correlation across many sensor readings, producing more robust results.

3.4.3 Resilience to Systematic Communication Failure

In order to test our algorithm's resilience to various types of failure, we ran several experiments where we systematically increase the number of such failures. We ran three different experiments: Host Failure, Bidirectional Link Failure, and Unidirectional Link Failure. In all experiments, the we added Gaussian error with a standard deviation of 20 percent of the total size of the temperature range to each sensor reading. We select this error model to illustrate the ability for our Graphical Modeling framework to compensate for extreme reading error caused by both the sensor itself and external properties imposed by the environment.

In the Mote Failure experiment, we increase the number of motes that are unable to communicate with all other motes. This removes all nodes in the graphical model owned by any "dead" hosts. We limit our error computation to nodes that are on live hosts. The results for the Mote Failure experiment are shown in Figure 3.5. The figure illustrates that while error grows gradually as hosts die, the standard deviation of error increases dramatically. As more motes die, they become much more reliant on their own readings, and thus are more susceptible to unreliable readings.

In the Bidirectional Link Failure experiment, we gradually sever an increasing number of links in the network connectivity graph. We currently restrict an edge in the graphical model to require a link in the network connectivity graph. Potentially, multicast routing and dynamic detection of link failures could overcome these errors, at the cost of more messages. Instead, severing a link between two hosts removes all edges in the graphical model for nodes on the first host that are connected to nodes to

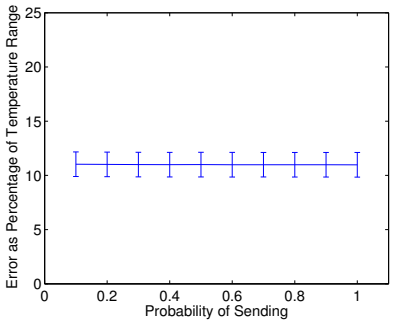


Figure 3.8. Average error as the probability of sending in a single iteration for SyncConstProb increases.

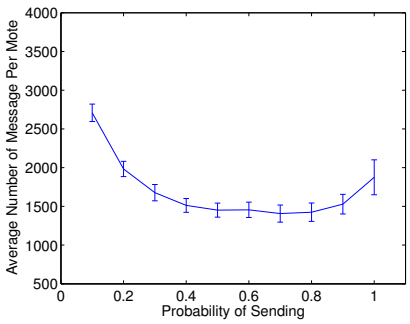


Figure 3.9. Number of messages sent as the probability of sending in a single iteration increases for SyncConstProb.

the other host, and vice versa. We show the results for this experiment in Figure 3.6, which shows that the error increases linearly as we sever links.

In the Unidirectional Link Failure experiment, we gradually cause more links in the network graph to only allow communication in one direction. This reflects asymmetric communication problems which are experienced in typical sensor-network deployments. The results in Figure 3.7 are nearly identical to those in Figure 3.6. Therefore, we conclude that our algorithm is resilient to unidirectional link failure. Note that this robustness is not a consequence of the smoothness and uniformity of the temperature estimation problem. Preliminary experiments with non-smooth and non-uniform MRFs where some edges correspond to much higher correlations compared to others, indicate similar behavior and graceful degradation under random link failures.

3.4.4 Resilience to Transient Failure

The proposed algorithm is very resilient to transient failures. We induce the equivalent of transient failures in the AsyncConstProb scheduling algorithm, whereby motes randomly do not transmit at every iteration. The magnitude of transience will

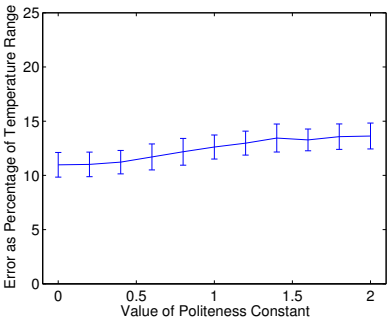


Figure 3.10. Average error as the exponent of the total variation difference between old and new messages increases for AsyncSmartProb.

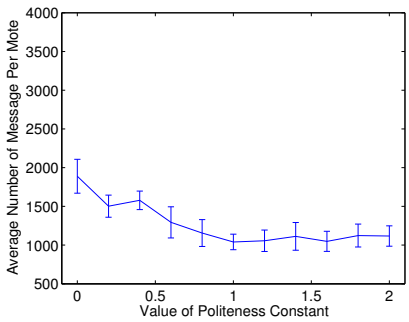


Figure 3.11. Number of messages sent as the exponent of the total variation difference between old and new messages increases for AsyncSmartProb.

just modify the number of iterations required before convergence. Thus, the algorithm does not need to rely on any robust messaging protocols. If a transmission does not get through, resending later will be sufficient for correctness.

3.4.5 Scheduling

For scheduling, we experiment with AsyncConstProb and AsyncSmartProb, evaluating the average error over all differences between the mean of the inferred random variable and the corresponding value of the diffusion process ground truth. We also evaluate how these algorithms affect the number of transmissions required for convergence.

As was previously discussed in the Section 3.4.4, and as illustrated in Figure 3.8, the error is not affected by changes in the probability that inter-mote messages are sent (or get through). Interestingly, the number of messages sent does not monotonically increase as the probability of sending increases. For very low probabilities, the important messages, which are needed for convergence, do not get transmitted often enough, and the number of iterations needs to be increased for the algorithm to converge. Therefore, even though the number of messages per iteration monotonically

decreases, the number of iterations increases at a rate that ends up increasing the total number of messages as can be observed in Figure 3.9. It is exactly this idea of *identifying and favoring the important messages that accelerate convergence* that leads to the investigation of AsyncSmartProb.

By changing the politeness constant η for AsyncSmartProb as described in Equation (3.1), we control the likelihood of transmission as a function of how informative the message is, as defined by the total variation distance. Setting $\eta = 0$, causes the algorithm to degenerates to AsyncAllTalk. As we increase η, we decrease the likelihood that the message will be transmitted for the same difference in messages between the last-sent and current message. Figure 3.10 illustrates that the amount of error incurred is minimal, while Figure 3.11 shows that we save approximately 50% of the messages, even compared to the minimum value of AsyncConstProb. Another interesting property is that the number of messages seems to be monotonically decreasing, as opposed to simple AsyncConstProb, which seems to get eventually send more messages even when decreasing the probability of sending. An advantage of AsyncConstProb is that modifying the politeness constant η directly trades time to convergence for the number of messages sent, while a sufficiently small choice of the probability p to send in AsyncConstProb can increase convergence time while also increasing messages sent.

3.4.6 Deployment Experiment

In our testbed, we inferred the temperature of a room in unobserved locations through RBP. We reduced the size of our graphical model to be a 6x6 node lattice, with each 2x2 section assigned to a host and the mote's true location again corresponding to the top-left node of its section. This setup yielded a deployment of 9 motes in a 3x3 grid where each mote was 6 inches apart from its closest neighbors

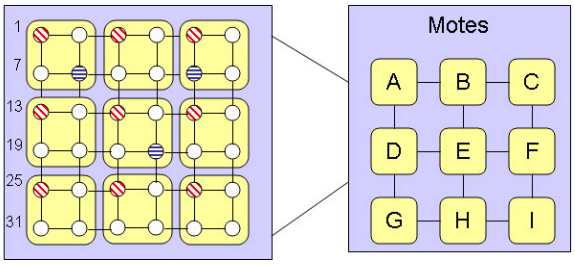


Figure 3.12. This figure illustrates the two layers of the Graphical Model (left) and Motes (right) which we used for our in-lab experiment. The red angle-slashed dots denote observation nodes, ie locations where we have sensor readings, and the blue-horizontal nodes denote locations where we logged unobserved temperature readings to determine error.

as we show in Figure 3.12. To create temperature gradients, we used two 1500 watt space heaters. To calculate our initial compatibility functions, we used the same 9 motes, but at a distance of 3 inches.

Similarly to the simulated experiments, we placed the heaters in 10 different setups, and used the collected temperature data to determine the compatibility functions according to equations (3.2), (3.3), and (3.4). We then ran our test by placing both space-heaters in the top right corner of our lab, and running Belief Propagation to infer the temperatures of the unobserved nodes. We compare the inferred temperatures at unobserved locations with the temperatures determined using 3 additional "spying" motes that collected direct measurements during the experiment, but were not involved in any information provided to the 9 Belief Propagation motes.

We plot the distributions of the nodes at the locations of the three spying motes in Figures 3.13, 3.14, and 3.15, which are discretized as histograms. We can see that the confidence of motes in their unobserved variables varies greatly, for instance there is very low variance in Node 8 (Figure 3.13), while significantly more variance in Node 11 (Figure 3.14). The red bar indicates the bucket in which the ground truth lies. The figures illustrate that the mean of this distribution well approximates the actual sensor reading. Interestingly, the distribution for Node 11 is bimodal. This is because

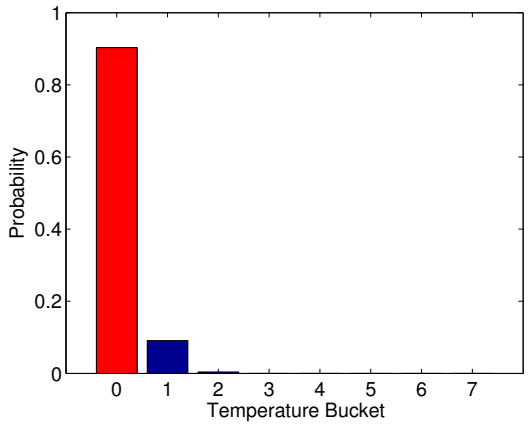


Figure 3.13. This is the histogram of the inferred distribution on Node 8 of our graphical model. The red bar is the bucket for the actual sensor reading.

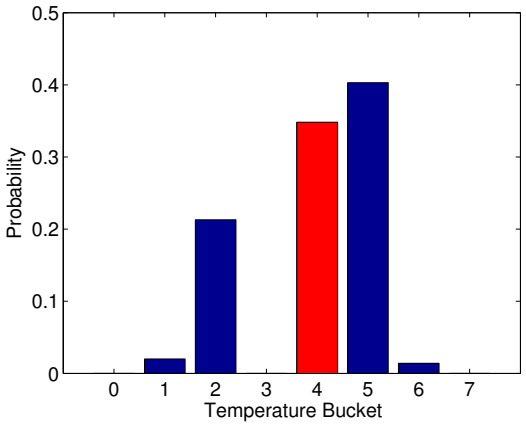


Figure 3.14. This is the histogram of the inferred distribution on Node 11 of our graphical model. The red bar is the bucket for the actual sensor reading.

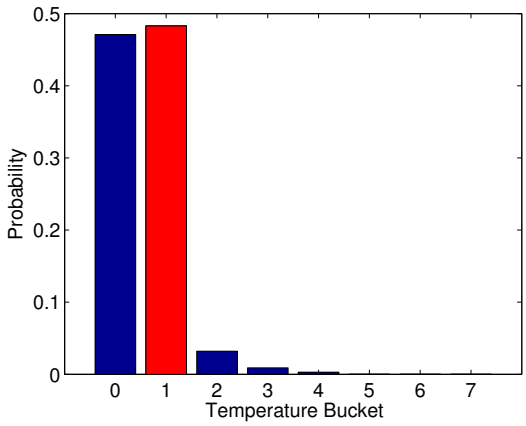


Figure 3.15. This is the histogram of the inferred distribution on Node 22 of our graphical model. The red bar is the bucket for the actual sensor reading.

when learning the model, the system never observed adjacent nodes both resolving to a temperature bucket of three. In this test, one of the readings next to Node 11 was 3, which resulted in the bimodal distribution. Table 3.1 shows the actual and

Node	actual temp	mean of marginal
8	24.5°C	23.2°C
11	43.6°C	41.4°C
22	27.5°C	25.3°C

Table 3.1. Comparison between actual and inferred temperatures for three nodes in the graphical model

inferred temperatures for these three nodes. The inference is correct to within 2.2°C for the three "spy" nodes. Lastly, we ran experiments to validate our simulations. We ran inference on the mote deployment with the model we learned from the real temperature readings. We then ran inference in simulation using the same graphical model and the sensor readings obtained from the motes. The results were identical to four significant figures.

3.5 Conclusions and Future Work

We presented a general architecture for using message passing algorithms for inference in sensor networks, using reweighted Belief Propagation. We demonstrate that RBP is robust to communication and node failures and hence constitutes an effective fit for sensor network applications. The robustness of our architecture is demonstrated in simulations and real mote deployment. An important feature of the proposed scheme is that it does not rely on a network layer to provide multi-hop routing and that our architecture provides meaningful results even when the motes experience severe noise in measurements or link failures. Note that, even though we show theoretically that any graphical model can be mapped to motes without requiring routing, in practice, some long-range correlations might introduce additional variables. This can be circumvented by simply ignoring the long-range links. In our temperature experiments, we found that no such long-range correlation edges existed.

We therefore believe that our architecture will be useful for many applications that involve statistical inference or data uncertainty in sensor networks.

Currently, our architecture does not consider dynamic models that exploit time-correlations. This extension could be useful for applications such as tracking and our work can be extended to exploit such temporal correlations. Addressing mote mobility is another issue we plan to explore especially the implications it might have in the dynamic mapping of nodes to motes.

Chapter 4

Safety: Dense Perceptive Pallets

4.1 Introduction

Effectively managing materials stored in warehouses is a multi-billion dollar global challenge. Known methods monitor entry and exit of materials, but do not monitor locations of materials. Often, materials are subject to regulatory requirements in terms of volumetric and spatial constraints. For example, US regulations for bio-pharma production require more than 1000 gallons of liquid corrosive (acid) material cannot be stored in one warehouse, and that an acid cannot be stored within 20 feet of a base and a flammable material cannot be stored within 20 feet of an oxidizer. Maintaining these constraints during transit and storage is a major challenge.

We consider the use of wireless sensor networks to automatically track "perceptive pallets" of materials in a warehouse to monitor such constraints. The system is required to operate dynamically in real-time, so that whenever a constraint violation is detected, the system can inform human operators, for instance by sounding an alarm. We use simulation to study the effects of mote placement: position error as

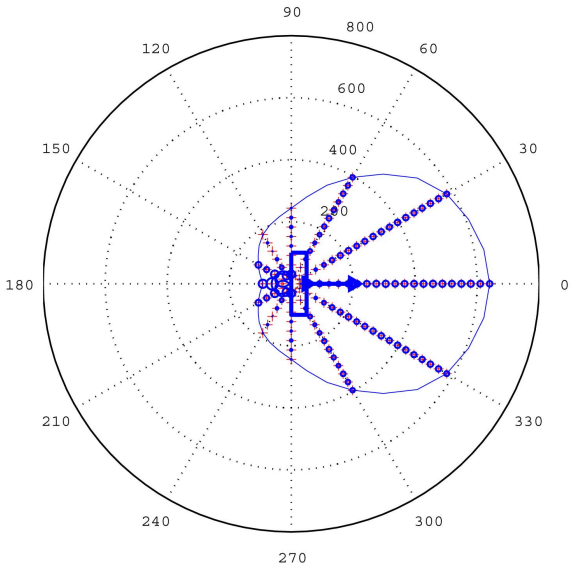


Figure 4.1. Measured ultrasound response patterns overlaid with best fit asymmetric (cardioid) response models. Rectangle and arrow depict the orientation of the ultrasonic microphone. Measured position errors are asymmetric and increase with angular distance from the positive x-axis.

a function of ceiling height and beacon density. We then perform experiments to characterize the effectiveness of asymmetric observation models.

One form of wireless sensor technologies for localization combine ultrasound bursts (chirps) with radio, and compare the time difference of arrival (TDOA) of the signals to estimate distances. Currently off-the-shelf commercially available motes include Cricket [134][152], Mica2 [81], and Telos [151]. However, the distance measurements produced by these motes are noisy due to hardware limitations and environmental effects.

In our initial study [42], we evaluated the ranging capability of the Cricket Mote, and discovered that the accuracy of the sensors greatly depends on relative angle between the transmitting microphone and listening ultrasonic sensor. The experimental results of that study are illustrated in Figure 4.1. In this chapter, we use the measured sensitivity pattern to develop new statistical observation models that take into account this angular bias, and show that they reduce spatial uncertainty

in position estimates. Experiments suggest that asymmetric observation models can improve position estimates by as much as 11%.

4.2 Related Work

While distributed sensor tracking is a topic of much research, as discussed in Chapter 2, we feel it is prudent to highlight specific work relating to supply chain management and the properties of the Cricket Motes used in our experiments.

4.2.1 Monitoring of Spatial Constraints for Warehouse Management

Research of warehouse management has mostly focused on optimal deployment and efficient retrieval [45] according to the principles of supply chain management. One of the promising technologies is Radio-Frequency Identification (RFID), which has been introduced to inventory management and storage tracking [131]. Paper [78] discusses the possibility of using RFID tags to improve the localization of mobile robots, where the robot carries two overhead RFID antennas to get distance readings relative to pre-deployed RFID tags and then estimates its location from maps of the environment. An operational RFID system consists of smart-chip tags embedded with product information coupled with RFID readers. The system must be centrally operated with state-of-the-art technologies.

A similar constraint-based distance estimation method has been proposed in [21], where a spatial distance graph is constructed by labeling nodes with distance constraints. Distance intervals were derived by performing a modified Floyd-Warshall shortest path iteration algorithm for all edges. This could be a promising approach for distance violation in warehouse management, since it does not involve a com-

plicated localization algorithm. However, perfect ranging results were assumed to implement the method, and each node must have enough memory to store the global graph and distance constraints information.

4.2.2 Pros and Cons of Cricket Motes

The Cricket location system [153] developed at MIT utilizes two types of signals, radio and ultrasound, and infers distance using the time difference of arrival of these two signals. The Cricket system implements compensation for the effects of temperature on the speed of sound, interference avoidance for arrival of radio signal and ultrasonic signals received from different transmitters, communication scheduling to avoid collision, and Kalman filtering for discarding incorrect distance samples.

In the Cricket localization algorithm, trilateration is utilized to derive the position of sensor nodes in [153], where a robust quadrilateral is introduced as a clique to avoid flip ambiguities. Additionally, signal processing in static environments using a least-square-error trilateration algorithm is used to infer position from multiple sensor readings. For a mobile environment, Kalman filters were used to derive the location. However, the selective directional sensitivity patterns of ultrasonic sensor motes make the measurements too biased, so that trilateration may be inconsistent if some of the distance readings are too noisy. Although the effects on accuracy caused by distance and angle were observed in [153], along with the existence of a cardioid sensitivity pattern, the authors did not explicitly compensate for these observations.

The most commonly recognized type of sensitivity beam patterns seen in ultrasonic sensors has one main lobe and several side lobes [37][124]. For the purposes of this chapter, we abstract the sensitivity pattern to a cardioid shape without side lobes. In [16] an observation model that accounts for the sensitivity pattern of ultrasonic receivers is presented. The model incorporates angle and distance measurements;

however, the sensitivity pattern does not account for received signal behind the microphone, as observed in a cardioid sensitivity pattern.

We extend the previous research on localization using Cricket motes by explicitly modeling the effect of the directional sensitivity pattern of ultrasonic sensors and making use of a probabilistic localization algorithm to derive consistent positions from noisy distance measurements.

4.2.3 Probabilistic Localization

Localization within a Bayesian probabilistic framework has found widespread applications in robotics [193], and has been shown to have good performance. Moreover, in some recent studies [166, 182, 189, 109, 50], methods based on a probabilistic framework have been introduced to localize within a sensor network. Probabilistic inference is made by observing multiple sets of noisy evidence. It has been shown in [84], that mobile seeds can be exploited to improve the accuracy of localization, by using Sequential Monte Carlo localization method.

Kalman filtering (KF) and Extended Kalman filtering (EKF) have been shown to be effective tools in the context of localization. However, Kalman filtering is known to be optimal only when both the transition and observation models are linear Gaussian distributions. This makes Kalman filtering infeasible for pallet monitoring, since there exist non-Gaussian components in the measurement due to the ultrasonic sensitivity pattern as illustrated in Section 4.1. As opposed to Kalman and Extended Kalman filtering, Particle Filtering explicitly approximates a probability distribution as a sample set. The advantage of Particle Filtering is its capability to represent an arbitrary probability model and its flexibility in dealing with dynamic and distributed systems.

Particle Filtering is an implementation of Bayesian filtering based on a Sequential

Monte Carlo (SMC) method [194]. The features of the method are a probabilistic transition model between each state and a probabilistic observation model to estimate state by observed evidence. The key of Particle Filtering is the representation of the posterior distribution by a set of weighted samples (or particles), so that it can approximately represent any probability distribution. In [166], a Particle Filtering approach is used to track intruders by a number of infrared sensors with binary outputs. The authors equipped the room with an array of binary sensors, characterized the firing probability and refractory period of these sensors by conditional probability distributions (CPDF), and derived the posterior probability by observing the firing events.

It has been shown that the accuracy of Particle Filtering is related to the number of samples. Therefore, the computation complexity of Particle Filter is directly related to the number of samples, and the optimal number of particles is difficult to determine. A computational complexity analysis of Particle Filters can be found in [24]. In our experience, however, we have found the trade-off between these convergence times and accuracy to be more than acceptable for the proposed application.

4.3 Problem Statement

The objective of our system is to monitor the location of Cricket sensor motes in a room of known dimensions. This tracking must operate in real time, and it must accurately localize both mobile and stationary sensors. In our proposed global architecture, envisioned in [42], Cricket motes (referred to as Beacons) are statically mounted in known locations on the ceiling, facing downward. Cricket motes (referred to as Motes) are also affixed to pallets placed on the floor, facing upward. Refer to Figure 4.2 for a visual depiction of the described setup.

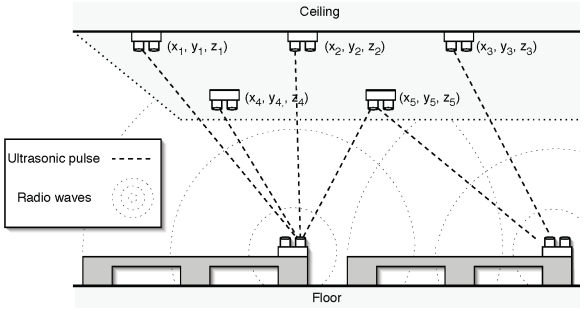


Figure 4.2. Depiction of layout of sensors for monitoring the location of hazardous materials in warehouses. The Cricket motes on the ceiling are referred to as Beacons, while the Cricket motes affixed to the pallets on the floor are referred to as Motes.

Each Mote broadcasts both an ultrasound pulse and a radio message containing a unique Mote identifier and a monotonically increasing message number at a predetermined rate managed by an internal timer. The message number is used by the Beacons to differentiate between messages and thus no global synchronization is required. Every Beacon that receives a radio message and ultrasound pulse computes its estimated distance from the Mote based on the difference in arrival time of the two signals. The distance estimates computed by all Beacons are centrally aggregated, and our algorithm uses these estimates to localize the Motes in discrete time.

4.3.1 Assumptions

It is assumed that the floor and ceiling are both flat and relatively level, and that all pallets are flat and of a uniform height. Other than the Beacons, Motes, and pallets, there are no other objects in the room. There is a central computer that is able to communicate directly with each ceiling-mounted Beacon; this is where the algorithm is run. It is also assumed that all pairwise distance estimates are probabilistically independent of each other.

4.3.2 System Input

The system is initialized with the Euclidean coordinates of each Beacon. The dimensions of the room are also specified by length, width, and height. The system collects pairwise distance estimates between some subset of all existing {Beacon, Mote} pairs at a predefined rate.

4.3.3 System Output

The system computes the estimated Euclidean coordinates of each Mote, with the objective of minimizing error defined as Euclidean distance between estimated and true position.

4.4 Using Particle Filtering for Localization

Table 4.1 contains important notation that will be used in the following sections. In our framing of the localization problem, we run a separate Particle Filter for each individual Mote. An object's state at time t, x_{it}, refers to the true Euclidean coordinates of mote i. The observations collected at time t, $\mathbf{z_{it}}$, refer to a vector of distance estimates computed by the beacons between time $t-1$ and time t, relative to mote i. Our transition model, $\mathrm{P}(\hat{x}_{ist}|\hat{x}_{is(t-1)})$, adds Gaussian noise to the Euclidean coordinates of $\hat{x}_{is(t-1)}$. The initial distribution of samples, $\mathrm{P}(\hat{x}_{is0})$, places sample states uniformly at random within the defined space. Our observation model, $\mathrm{P}(\mathbf{z_{it}}|\hat{x}_{ist})$, gives the likelihood of the computed distance estimates based on empirical data gathered about the sensitivity pattern of the Cricket motes.

4.4.1 Deriving the Observation Model

It was demonstrated in [42] that the pairwise distance estimates produced by Cricket motes vary based on the relative angle of the two motes. By the independence assumption of distance estimates, the observation model is expressed as the following:

$$\mathrm{P}(\mathbf{z_{it}}|\hat{x}_{ist}) = \prod_{j=1}^{m} \mathrm{P}(z_{ijt}|\hat{x}_{ist}) \tag{4.1}$$

Because the locations of all Beacons B are known, given a sample state $\hat{x}_{ist}$ we can compute the Euclidean distance and relative angle to each individual Beacon j. Let (x_i, y_i) represent the state $\hat{x}_{ist}$, let (x_j, y_j) represent the Euclidean coordinates of Beacon j, and let h represent the height of the ceiling to which j is mounted. For each pair $\{\hat{x}_{ist}, j\}$, we can compute $\hat{d}_{ijst}$ and $\hat{\theta}_{ijst}$ as follows:

$$\hat{d}_{ijst} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + h^2} \tag{4.2}$$

$$\hat{\theta}_{ijst} = \arccos\left(\frac{h}{\hat{d}_{ijst}}\right) \tag{4.3}$$

Using these two aspects of a Mote's state, we can now express our observation model as the following:

$$\mathrm{P}(\mathbf{z_{it}}|\hat{x}_{ist}) = \prod_{j=1}^{m} \mathrm{P}(z_{ijt}|\hat{d}_{ijst}, \hat{\theta}_{ijst}) \tag{4.4}$$

4.4.2 Using Experimental Data to Model Acoustic Sensitivity

During the calibration phase, we fit the observation model to the underlying truth from experimental data. Calibration is performed once before the Particle Filtering

algorithm is executed. From the results in [42], we can examine the distance estimate z_{ij} produced between a Mote i and Beacon j based on their true relative distance d_{ij} and angle θ_{ij}. In these experiments, the distance estimates z_{ij} were collected for a variety of $\{d_{ij}, \theta_{ij}\}$ pairs.

Because the acoustic sensitivity pattern of the ultrasound signal exhibits a cardioid shape, as depicted in Fig 4.1, we assume that the mean, μ, and the variance, σ^2 can be reasonably approximated as a polar function of any arbitrary distance and angle pair $\{d, \theta\}$. We refer to this observation model as Asymmetric Polar (AP):

$$\mu(d,\theta) = ad\cos\theta + bd\sin\theta + c \tag{4.5}$$

$$\sigma^2(d,\theta) = pd\cos\theta + qd\sin\theta + r \tag{4.6}$$

Using linear regression, we computed the specific values for the coefficients a, b, c, p, q, r that provided the least squared error over all pairs $\{d_{ij}, \theta_{ij}\}$ for which there was available experimental data. By preprocessing these coefficients, much of the computation is implicitly performed by the formulation, rather than at run time. Given these coefficients, we can compute $\mu(d,\theta)$ and $\sigma^2(d,\theta)$ for any $\{d,\theta\}$. These μ and σ^2 functions can then be substituted into the standard Gaussian probability density function. Thus, the probability distribution given in Equation (4.4) can be approximated using the following function, instantiated with actual $\{\hat{d}_{ijst}, \hat{\theta}_{ijst}\}$ values:

$$\mathrm{P}(z_{ijt}|\hat{d}_{ijst}, \hat{\theta}_{ijst}) = \frac{1}{\sqrt{2\pi\sigma^2(\hat{d}_{ijst},\hat{\theta}_{ijst})}} \exp\left\{-\frac{(\mu(\hat{d}_{ijst},\hat{\theta}_{ijst})-z_{itj})^2}{2\sigma^2(\hat{d}_{ijst},\hat{\theta}_{ijst})}\right\} \tag{4.7}$$

To compare the formulation as a polar function, we constructed a similar observation model with μ, and σ^2 formulated as linear functions of d and θ. This observation model is referred to as Asymmetric Linear (AL). For this model, Equations (4.5) and

(4.6) are formulated as follows:

$$\mu(d,\theta) = ad + b\theta + c \tag{4.8}$$

$$\sigma^2(d,\theta) = pd + q\theta + r \tag{4.9}$$

We then computed this set of coefficients a, b, c, p, q, r using the same approach as described above. Using these coefficients and the functions in Equations (4.8) and (4.9), μ, and σ^2 can be used in Equation (4.7) without additional modification.

Finally, to test the effect of the relative angle of mote poses when computing distance estimates, we also constructed an observation model with μ, and σ^2 formulated as linear functions of only distance d. This observation model is referred to as Symmetric Linear (SL). For this model, Equations (4.5), and (4.6) are formulated as follows:

$$\mu(d,\theta) = ad + 0\theta + c \tag{4.10}$$

$$\sigma^2(d,\theta) = pd + 0\theta + r \tag{4.11}$$

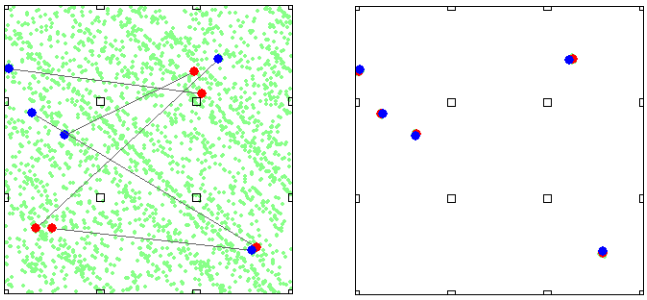
Again, this set of coefficients a, b, c, p, q, r was computed using linear regression with $b = q = 0$. Using these coefficients and the functions in Equations (4.10) and (4.11), μ, and σ^2 can be used in Equation (4.7) without additional modification. The accuracy with which each of these models localized the Motes is evaluated in Section 4.5.2.

4.5 Experimental Results

We evaluate the performance of our algorithm with simulator and physical experiments. The objective of the simulator experiments is to determine an acceptable

physical beacon configuration. To determine which configurations are acceptable, we look for a layout suitable for a warehouse environment that has estimation error less than 10 cm. We define estimation error ($|x_{it} - \hat{x}_{it}|, \forall i \in M, t \in T$) as an objective in Section 4.3.3.

4.5.1 Simulation Results



(a) Initial Distribution (b) After Convergence

Figure 4.3. Graphical User Interface of Particle Filter simulator. (a) Initial particle distribution of first iteration. (b) Estimated positions from of the motes. The blue dots represent the actual poses of motes, red dots for the estimated ones, squares for beacons and green dots for particles. Though difficult to see, particles are present under the estimated positions in (b). The length of the line connecting the actual motes and estimated nodes reflect the errors of estimation.

To describe the best physical configuration of beacons, we visualize estimation error according to the dimensions of ceiling height and beacon density. We constructed a simulator that was capable of performing many trials in a short amount of time. Performance was measured as distance estimation error for various ceiling heights h (reported in cm) and inter-beacon (beacon to beacon) distances k (reported in cm). Beacons are evenly spaced at the inter-beacon distance in a grid. Our objective was to determine an appropriate beacon configuration in order to minimize total computation expense, total number of needed beacons at a reasonable ceiling height while still maintaining an acceptable measure of error.

The simulator is implemented in a Java 1.5 environment with an Applet Graphical

User Interface (GUI) used to visualize the actual positions, estimated positions, and particle locations of Beacons and Motes (shown in Figure 4.3). At initialization, a set of particles for each node is generated with a uniform distribution as shown in Figure 4.3(a). After computing the observed measurements, the posterior probability is calculated and the particles converge to an estimated position as shown in Figure 4.3(b). Each mote is given a random trajectory of (x_t, y_t) pairs. The trajectory is generated at each time step by perturbing the current, true position with a sample drawn from a Gaussian distribution with $(\mu = 0, \sigma = 10)$.

The observation model used for the experiments is Asymmetric Linear. Experiments were run on IBM xSeries 330 machines equipped 1.0 GHz Intel Pentium III CPUs and 1.5GB ECC PC133 SDRAM. Each experiment consisted of 10 motes, over 10 iterations, with 5000 particles per mote. These experiments had an average run time of 1.25 minutes.

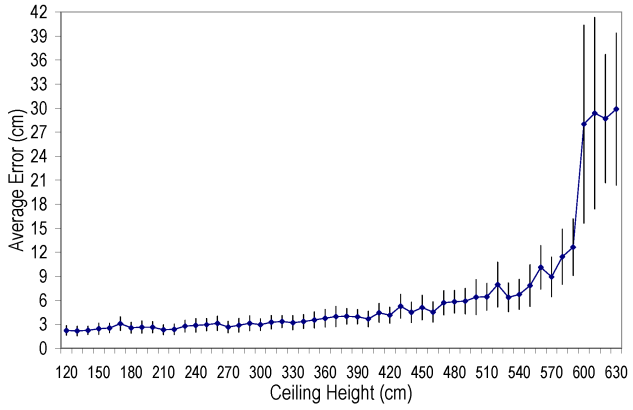


Figure 4.4. Estimation error against ceiling height. In this representative plot, inter-beacon distance $k = 180$ and particle density is 5000. Average error of all 10 nodes over 10 iterations is shown against each ceiling height. Standard deviation is shown with vertical bars. A sharp threshold is shown for $h > 590$.

For our first experiment, we evaluate ceiling height with moving nodes. Ten motes are placed in the simulator space of 1000x1000 cm size room. Beacon density is held constant and ceiling height is varied from $h = 120, \ldots, 640$ with a linear step

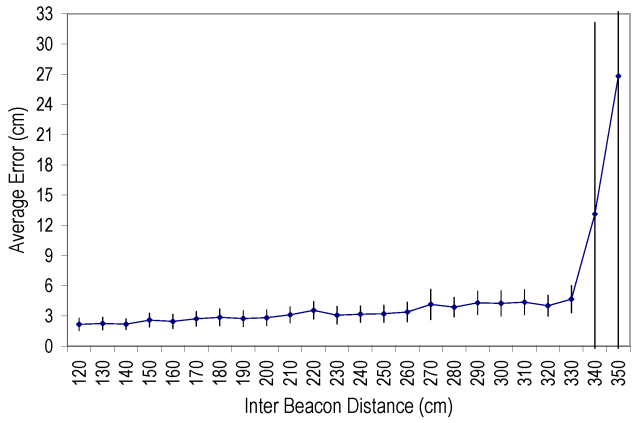


Figure 4.5. Estimation error against beacon density. In this representative plot, ceiling height $h = 240$ and particle density is 5000. Average error of all 10 nodes over 10 iterations is shown against each inter-beacon distance. Standard deviation is shown with vertical bars. A sharp threshold is shown for $k > 330$.

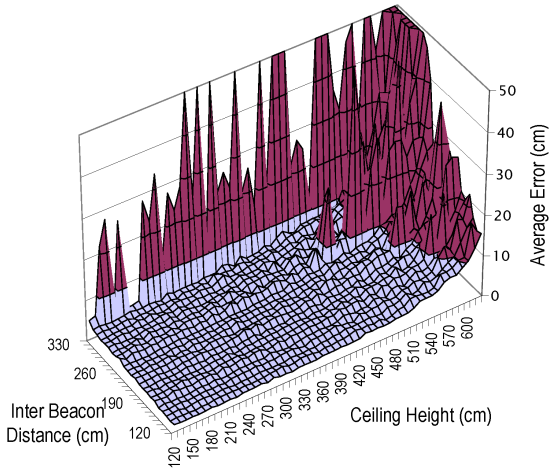


Figure 4.6. Estimation error along both dimensions studied for beacon configuration. This plot depicts an "acceptable zone" for beacon configurations. Any configuration with error less than 10 cm is deemed acceptable.

size of 10. This measurement is evaluated for all beacon densities as described in the next experiment. As shown in Figure 4.4, distance estimation error increases proportionally with ceiling height and is sharply thresholded for heights above 590 cm.

In the next experiment, we evaluate beacon density with moving nodes. Again, ten motes are placed in a simulator space of an 1000x1000 cm size room. Ceiling

height is held constant while inter-beacon distance is varied from $k = 120, \ldots, 350$ with a linear step size of 10. This measurement is evaluated for all ceiling heights evaluated in the previous experiment. As shown in Figure 4.5, distance estimation error increases proportionally with inter-beacon distance and is sharply thresholded for distances above 330 cm.

Both ceiling height and inter-beacon density are plotted in Figure 4.6. This plot depicts an "acceptable zone" of inter-beacon density and ceiling height. We define acceptable to be configurations where the error is less than 10 cm.

4.5.2 Physical Experiment Results

Using the configuration results of the simulator, we installed the motes in our lab to test performance. The model of the mote was MCS410CA (or Cricket mote v2), running TinyOS 1.1.7. Using the MIB510CA programming board, the Cricket software (release 2.3.0) was modified to transmit pairwise distances to a bay station. None of the on-board distance calculations were modified.

The primary metric of our experiment was again distance estimation error $|x_{it} - \hat{x}_{it}|, \forall i \in M, t \in T$. We installed six Cricket motes in beacon mode at a uniform height of 274 cm with an approximate inter-beacon distance of 198 cm. Beacons were installed parallel to the floor with ultrasonic sensors directed downward. Rotation parallel to the floor plane is not explicitly controlled, whereas tilt and yaw are avoided. A trajectory was then marked on the floor with the same x-y coordinate plane as the beacons. One cricket was configured as a listener and then moved along the trajectory on the floor.

In order to apply our algorithm to the sensor evidence, the motes were programmed to send pairwise distance estimates at a set interval to the bay station. The experiment is designed to control specific locations at specific iterations. Ac-

(a) Beacon installation (b) Tracked Mote

Figure 4.7. Installed cricket motes at a uniform height of 274 cm (a). Ultrasonic sensors are directed downward. Trajectory of the mote as steps are marked on the floor and measured to get the actual location (b). Mote to be localized is directed upward.

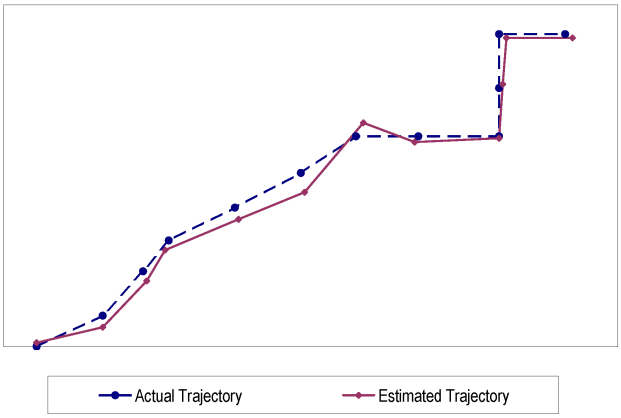


Figure 4.8. Top view of the lab floor. Lines indicate the actual versus estimated trajectory in a portion of the lab of 450 cm x 250 cm. The observation model for this experiment is Asymmetric Polar. The plot depicts one experimental run whose overall estimation error was closest to the mean reported for the AP observation model.

cordingly, each iteration is 10 seconds long. To process the algorithm, the recorded pairwise distances are passed into the simulator to analyze and visualize the estimated positions of the motes. A comparison of actual and estimated trajectory of the experimental run is shown in Figure 4.8.

We compare the effect on estimation error for the three observation models Asym-

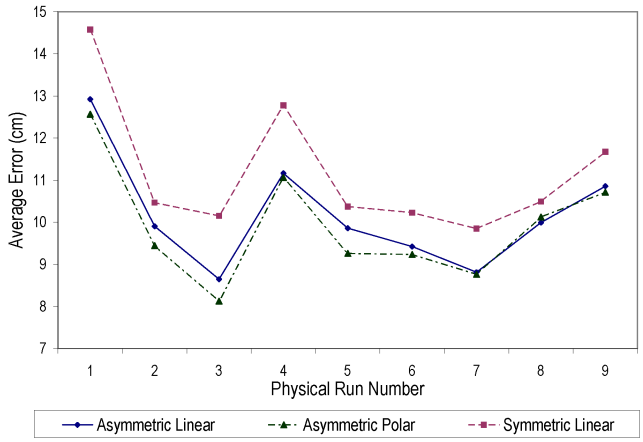


Figure 4.9. Comparison of the three observation models in this chapter. The pairwise distances of 33 iterations from the physical experiment are compared over 9 experimental runs of 50 Particle Filter estimations each. When angle is considered (AL and AP), error is shown to be strictly less than a distance only observation model.

metric Polar (AP), Asymmetric Linear (AL), and Symmetric Linear (SL). Using the same experimental data from the physical experiments, the three observation models are compared in Figure 4.9. It is shown that an asymmetric observation model (AL and AP) is strictly better than an observation model that does not consider angle (SL). Moreover, this experiment demonstrates that a linear approximation with angular dependence of the empirical data of the ultrasonic sensors (AL) has performance that is very close to that of an approximation that considers the polar model of the cardioid sensitivity pattern (AP). The estimation error for each model over all physical experiments is SL: $\mu = 4.40, \sigma = 4.50$, AL: $\mu = 4.01, \sigma = 4.06$, and AP: $\mu = 3.91, \sigma = 4.05$. These experiments suggest that asymmetric observation models can improve position estimates by as much as 11%.

4.6 Conclusion and Future Work

We consider the use of wireless sensor networks to automatically track "perceptive pallets" of materials in warehouses for the purpose of monitoring volumetric and spatial constraints. We measure and characterize the ultrasound response from standard "Cricket" wireless sensor motes and beacons. We develop a non-parametric Particle Filtering approach to estimate trajectories of moving motes and introduce two asymmetric observation models that incorporate measured cardioid-shaped response patterns of ultrasound. We use simulation to study the effects of mote placement: position error as a function of ceiling height and beacon density, and then perform physical experiments to evaluate the effectiveness of asymmetric vs. symmetric observation models for pallet tracking.

The key contribution of our research is to measure the effects of spatial asymmetry in ultrasonic sensors and to incorporate these sensitivity patterns into the Particle Filtering observation model.

In future work, we will incorporate the fact that we usually have a precise map of the warehouse interior. Since ultrasound requires line of sight communication, this map can account for some of these challenges by explicitly considering known obstacles, known transportation routes, and known storage areas. We will combine this information with the Particle Filter to further improve position estimates and develop fast algorithms for automatic monitoring of volumetric and spatial constraints. Additionally, we will expand the physical experimentation to an actual warehouse environment using the techniques and configurations recommended by this research.

Table 4.1. Dense Perceptive Pallet Notation

M	The set of Motes, $i \in M$.
B	The set of Beacons, $j \in B$.
S	The set of Samples, $s \in S$.
T	The set of Particle Filter iteration Times, $t \in T$.
h	The height of a mounted beacon, assumed to be a uniform constant.
x_{it}	The true state of Mote i at time t. State is defined by Euclidean (x, y) coordinates.
z_{ij}	The distance estimate between Mote i and Beacon j during calibration.
d_{ij}	The true Euclidean distance between Mote i and Beacon j during calibration.
θ_{ij}	The true relative angle between Mote i and Beacon j during calibration.
$\mathbf{z_{it}}$	The vector $[z_{i1t}, \ldots, z_{ijt}, \ldots, z_{imt}]$ of computed distance estimates relative to Mote i at time t, relative to each Beacon j, where m is the number of Beacons.
$\hat{x}_{it}$	The estimated state of Mote i at time t.
$\hat{x}_{ist}$	The state of sample s representing a possible state of Mote i at time t.
$\hat{d}_{ijst}$	The Euclidean distance between the sample state $\hat{x}_{ist}$ and Beacon j.
$\hat{\theta}_{ijst}$	The relative angle between the sample state $\hat{x}_{ist}$ and Beacon j.
a, b, c, p, q, r	Coefficients used for experimental data fitting.

Chapter 5

Safety: Sparse Perceptive Pallets

5.1 Introduction

Collaborative self-localization and tracking using wireless sensors has many applications such as tracking vehicles on roadways, or firefighters is burning buildings. These techniques can also be directly applied to our Perceptive Pallets project described in Chapter 4, tracking pallets in warehouses, as a mechanism for ensuring safety. In this chapter, we consider the problem of tracking multiple moving robots using noisy sensing of inter-robot and inter-beacon distances. In contrast to the dense perceptive pallets formulation in Chapter 4, the sparse perceptive pallets formulation in this chapter uses inter-distance readings between pallets (or more generally robots), as opposed to just using pallet communications with beacons. In our problem, sensing is local: there are three fixed beacons at known locations, so distance and position estimates propagate across multiple robots.

Consider a graph where nodes correspond to beacons or mobile robots, and edges link pairs of nodes for which distance measurements are available. Robots may be multiple hops away from beacons. This chapter relaxes the requirement from dense

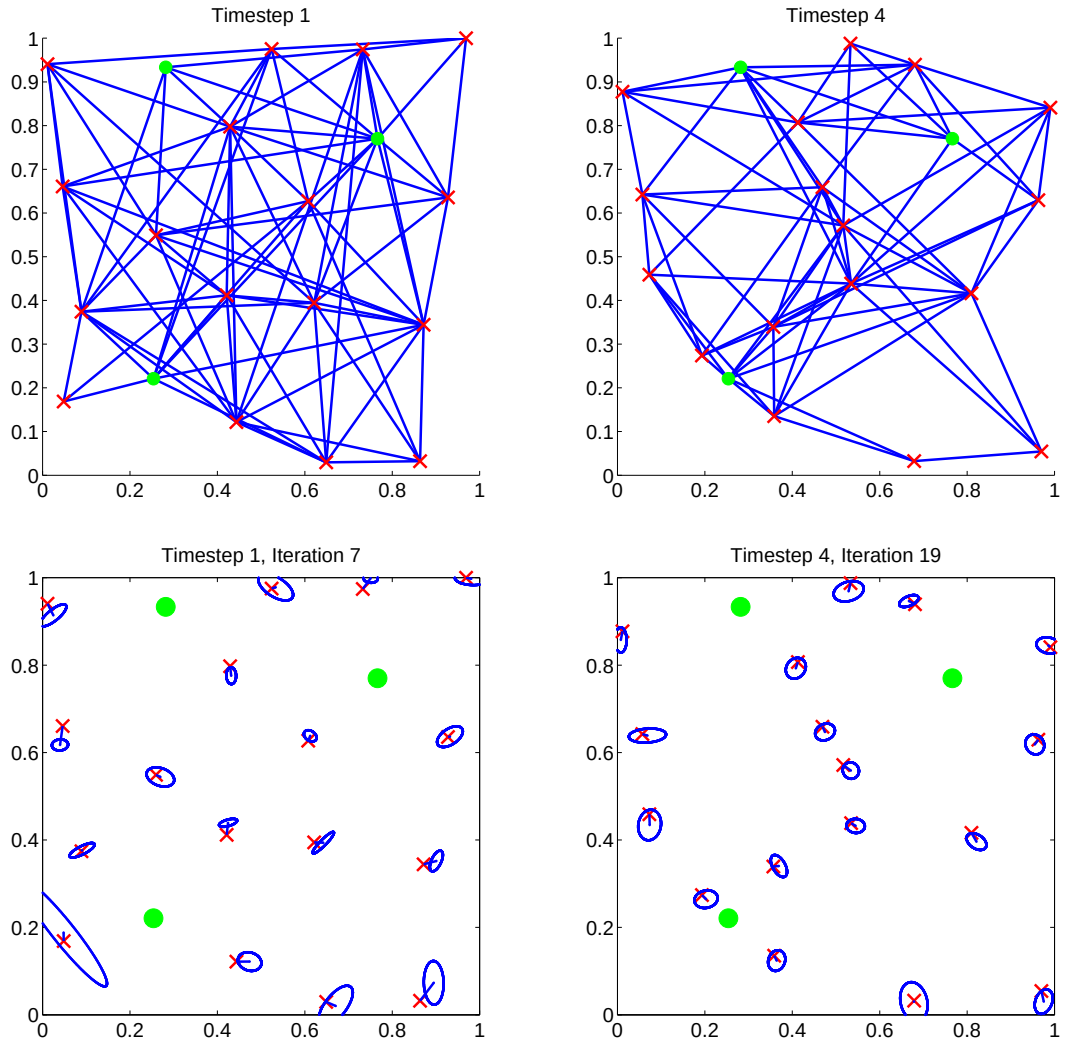


Figure 5.1. Tracking seventeen robots at first and fourth timestep. The top figures depict inter-distance sensor connectivity: edges link pairs of robots that share a distance reading. Three green discs denote fixed reference beacons. True robot locations are indicated with red crosses. The bottom figures overlay true robot locations with point estimates of robot location generated by the NBP algorithm, shown as blue circles (variance is not yet indicated). Note that although robots move substantially between timesteps and few robots are in direct contact with beacons, error in all cases is negligible.

Perceptive Pallets described in Chapter 4, where we assumed that beacons were placed with dense enough spacing such that beacon-only localization was possible, and robots did not need to communicate directly with one another. The inter-distance tracking problem is illustrated in Figure 5.1. Inter-distance tracking is also closely related to

simultaneous localization and mapping (SLAM) problems [191], in which each robot is treated as a uniquely identifiable, but mobile, landmark.

We formalize the inter-distance tracking problem using a probabilistic graphical model, which integrates prior estimates about beacon locations, sensor models, and robot dynamics. In contrast with previous localization methods for networks of mobile robots, our distributed tracking algorithm is formulated as a variant of the Belief Propagation (BP) [148] algorithm, known as nonparametric Belief Propagation (NBP) [183], for performing inference over the graphical model. It is used to infer globally consistent estimates of robot location from noisy, local distance measurements. We demonstrated the feasibility of running BP in a real network of MICA2 motes for static sensor fusion in Chapter 3. NBP approximates posterior distributions of unobserved variables by sets of representative samples. It can be seen as a generalization of Particle Filters [191] to domains with richer, non-temporal structure. NBP does not require linear or Gaussian models [95] and can model multi-modal and ring-shaped distributions produced by distance sensors.

In principle, discretized approximations of BP based on a regular spatial grid are also possible. For an $M \times M$ grid and n robots, however, storage and communications requirements scale as $O(nM^2)$, and computation as $O(nM^4)$ assuming a maximum number of neighbors. Efficiency improvements have been suggested in certain special cases [55, 215], but cannot be directly applied to the graphical model underlying inter-distance tracking. In contrast, NBP directly leads to a distributed, message-passing algorithm which can efficiently scale to large spatial domains.

The algorithm runs in two phases. In phase I, we localize robots at the first timestep, using a similar formulation to the static localization algorithm of Ihler et al. [88] and described in Section 5.3. Section 5.4 describes phase II, in which a dynamics model is used to combine inter-distance measurements over time.

Our tracking algorithm uses dynamics models to improve accuracy, while reducing computation and communication requirements by up to 3 times when compared to a localization-only approach for a 20 robot network. In our experiments, we demonstrate examples where our tracking algorithm determines the location of the robot, while an approach processing each timestep independently fails to do so. Other approaches, for example those based on directly weighting temporal samples by inter-distance likelihoods, failed to localize the robots. Our approach, based on a novel decomposition of the temporal dynamics model, allows us to integrate multi-hop inter-distance readings in a single timestep and track the robots. We also show how NBP message updates should be scheduled to achieve robust, reliable convergence.

In the Experiments section, we compare our algorithm to a Monte-Carlo approach developed by Dil et al. [46]. We find that our algorithm is slower but 3x to 4x times more accurate, and more robust to noisy inter-distance sensor readings.

5.2 Problem Formulation

5.2.1 Input, Assumptions, and Output

Our models supports tracking applications where n robots move around a space for T timesteps. The location of robot s at time τ is denoted by $x_{s,\tau} \in \mathbb{R}^2$. We currently model the space as a 1x1 unit square, but our approach can be easily relaxed to support other geometries. Each robot is equipped with sensors for inter-distance measurement. We denote the (noisy) distance estimate between robots s and t at time τ as $d_{st,\tau}$. As distance estimates are based on signals emitted by nearby robots, the likelihood a sensor will detect a nearby sensor diminishes with distance. We refer to the subset of robots which receive distance readings from robot s at time τ as the neighbors of s, and denote them by $\Gamma(s, \tau)$.

Our objective is to determine position estimates $\hat{x}_{s,\tau}$ which minimize the mean squared error of each robot's location estimates, averaged over all time steps:

$$L(x, \hat{x}) = \frac{1}{nT} \sum_{\tau=1}^{T} \sum_{s=1}^{n} ||\hat{x}_{s,\tau} - x_{s,\tau}||^2 \tag{5.1}$$

The NBP algorithm provides non-parametric distributions at each time step that can be used to quantify confidence based on variance. A multi-modal distribution can be resolved in a variety of ways, as discussed in the Future Work section.

5.2.2 Modeling System Properties

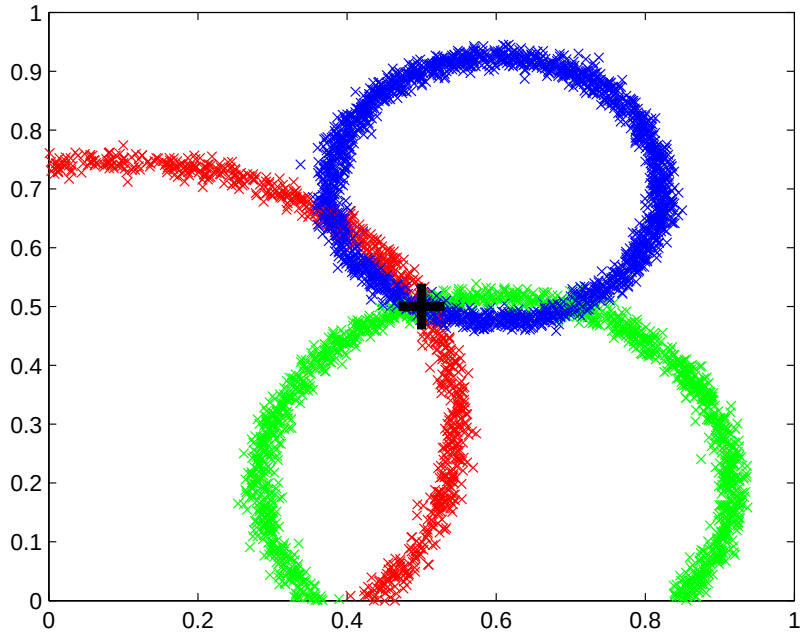


Figure 5.2. This figure illustrates three ring-shaped distributions produced by inter-distance readings between three beacons and a robot. We represent the location of the robot with a large black +.

The state of each robot s at time τ is represented by its position and velocity $\chi_{s,\tau} = (x_{s,\tau}, \dot{x}_{s,\tau})$. We model errors in the estimated distance between robots s and t

as follows:

$$d_{st,\tau} = ||x_{s,\tau} - x_{t,\tau}|| + \nu_{st,\tau} \qquad \nu_{st,\tau} \sim \mathbb{N}(0, \sigma_\nu^2) \tag{5.2}$$

To unambiguously localize a robot, inter-distance readings from at least three other localized robots or beacons are required. We demonstrate three ring-shaped distributions from exact beacons, used to localize a single robot in Figure 5.2. For localization at the first timestep, the network must thus contain at least three beacons, and all robots must have at least three inter-distance readings.

Following [136], we model the decay in a robot's probability P_o of measuring the distance to another robot as follows:

$$P_o(x_{s,\tau}, x_{t,\tau}) = \exp\left\{-\frac{1}{2}||x_{s,\tau} - x_{t,\tau}||^2/R^2\right\} \tag{5.3}$$

Here, R is a parameter specifying the range of the transmitter used for distance estimation. More elaborate models could be used to capture environmental factors or multi-path effects.

Each robot moves over time according to a dynamics model $P(\chi_{s,\tau+1} \mid \chi_{s,\tau})$ defined as follows:

$$x_{s,\tau+1} = x_{s,\tau} + \dot{x}_{s,\tau+1} \tag{5.4}$$

$$\dot{x}_{s,\tau+1} = \dot{x}_{s,\tau} + \omega_{s,\tau} \qquad \omega_{s,\tau} \sim \mathbb{N}(0, \sigma_\omega^2) \tag{5.5}$$

In practice, these velocities are often unobserved variables used to explain the typically smooth character of real robotic motion. Alternatively, sensors such as accelerometers could provide more precise dynamics information.

5.2.3 Decomposing Gaussian Mixtures

In the tracking algorithm developed in Section 5.4, it is necessary to decompose a Gaussian mixture $p(\chi) = p(x, \dot{x})$ into marginal and conditional distributions

$p(x)p(\dot{x} \mid x)$. Because the conditional distribution of a subset of jointly Gaussian variables is Gaussian, conditional distributions of Gaussian mixtures are mixtures of lower-dimensional Gaussians. We now derive formulas for the marginal and conditional portions of each mixture component. As the same formulas apply to all components, we drop the $\cdot^{(i)}$ notation.

Consider a Gaussian mixture component with weight w_χ, and mean vector and covariance matrix

$$\mu_\chi = \begin{bmatrix} \mu_x \\ \mu_{\dot{x}} \end{bmatrix}, \qquad \Sigma_\chi = \begin{bmatrix} \Sigma_{x,x} & \Sigma_{x,\dot{x}} \\ \Sigma_{\dot{x},x} & \Sigma_{\dot{x},\dot{x}} \end{bmatrix}. \tag{5.6}$$

The marginal $p(x)$ has weight w_χ, mean μ_x, and covariance matrix Σ_x. The conditional density $p(\dot{x} \mid x = a)$ then has the following mean, covariance, and weight:

$$\mu_{\dot{x}|x} = \mu_{\dot{x}} + \Sigma_{\dot{x},x}\Sigma_{x,x}^{-1}(a - \mu_x) \tag{5.7}$$

$$\Sigma_{\dot{x}|x} = \Sigma_{\dot{x},\dot{x}} - \Sigma_{\dot{x},x}\Sigma_{x,x}^{-1}\Sigma_{x,\dot{x}} \tag{5.8}$$

$$w_{\dot{x}|x} \propto w_\chi \left(\frac{|\Sigma_{\dot{x}|x}|}{|\Sigma_\chi|} \right)^{\frac{1}{2}} \exp\left\{ -\frac{1}{2}\sigma_0^T \Sigma_\chi^{-1} \sigma_0 \right\} \tag{5.9}$$

The transformations from a single joint multivariate Gaussian to a single conditional multivariate Gaussian, transforming the mean and standard deviation, are determined by standard formulas[85]. Our formulas look slightly different, as we have computed $P(\dot{x}|x)$, and the direct application of the transform would compute $P(x|\dot{x})$. The weight $w_{\dot{x}|x}$ depends on a centered observation vector:

$$\sigma_0 = \begin{bmatrix} a - \mu_x \\ \mu_{\dot{x}|x} - \mu_{\dot{x}} \end{bmatrix} \tag{5.10}$$

We use this decomposition to factorize dynamics messages sent between the same robot at subsequent timesteps. By first integrating distance readings via the smaller position-only marginals, and then drawing appropriately reweighted velocity samples, we can integrate multi-hop inter-distance information over time.

Proposition 3. *The conditional of a multi-modal Gaussian should be weighted by*

$$w_{\dot{x}|x} \propto w_\chi \left(\frac{|\Sigma_{\dot{x}|x}|}{|\Sigma_\chi|}\right)^{\frac{1}{2}} \exp\left\{-\tfrac{1}{2}\sigma_0^T \Sigma_\chi^{-1} \sigma_0\right\}$$

Proof.

$$P(X, \dot{X}) = P(\dot{X}|X)P(X)$$

Because $P(X)$ is just a normalizing factor relating the conditional to the joint, we can rewrite the distribution as

$$P(X, \dot{X}) = P(\dot{X}|X)w$$

Because w will vary only as a function of a, we can simplify our equations by solving for w at $[\mu_{\dot{x}|x}, a]^T$ for the joint and just $\mu_{\dot{x}|x}$ for the conditional, as we already condition that that position is at state a.

$$\begin{aligned} w &= P(X, \dot{X})/P(\dot{X}|X) && (5.11) \\ &= \mathbb{N}\left(\begin{bmatrix} a \\ \mu_{\dot{x}|x} \end{bmatrix}, \mu_\chi, \Sigma_\chi\right) / \mathbb{N}\left(\mu_{\dot{x}|x}, \mu_{\dot{x}|x}, \Sigma_{\dot{x}|x}\right) && (5.12) \end{aligned}$$

By expanding the equations out, we see

$$\begin{aligned} w &= \frac{\frac{1}{(2\pi)^2|\Sigma_\chi|^{1/2}} \exp\left\{-\frac{1}{2}\left(\begin{bmatrix} a \\ \mu_{\dot{x}|x} \end{bmatrix} - \mu_\chi\right)^T \Sigma_\chi^{-1} \left(\begin{bmatrix} a \\ \mu_{\dot{x}|x} \end{bmatrix} - \mu_\chi\right)\right\}}{\frac{1}{(2\pi)|\Sigma_{\dot{x}|x}|^{1/2}} \exp\{-\frac{1}{2}(\mu_{\dot{x}|x} - \mu_{\dot{x}|x})^T \Sigma_{\dot{x}|x}^{-1}(\mu_{\dot{x}|x} - \mu_{\dot{x}|x})\}} && (5.13) \\ &= \frac{|\Sigma_{\dot{x}|x}|^{1/2}}{(2\pi)|\Sigma_\chi|^{1/2}} \exp\left\{-\frac{1}{2}\begin{bmatrix} a - \mu_x \\ \mu_{\dot{x}|x} - \mu_{\dot{x}} \end{bmatrix}^T \Sigma_\chi^{-1} \begin{bmatrix} a - \mu_x \\ \mu_{\dot{x}|x} - \mu_{\dot{x}} \end{bmatrix}\right\} && (5.14) \end{aligned}$$

alternatively,

$$w = \frac{|\Sigma_{\dot{x}|x}|^{1/2}}{(2\pi)|\Sigma_\chi|^{1/2}} \exp\left\{-\frac{1}{2}\begin{bmatrix} a - \mu_x \\ \Sigma_{\dot{x},x}\Sigma_{x,x}^{-1}(a - \mu_x) \end{bmatrix}^T \Sigma_\chi^{-1} \begin{bmatrix} a - \mu_x \\ \Sigma_{\dot{x},x}\Sigma_{x,x}^{-1}(a - \mu_x) \end{bmatrix}\right\} \quad (5.15)$$

■

5.3 Phase I: Localization Algorithm

The goal of the inter-distance localization sub-problem is to infer the location of each node using only information from a single timestep (the first). Our formulation closely follows the algorithm proposed by Ihler et al. [88]. Since we consider a single timestep, we drop the τ notation and velocity state variables, and denote the position of robot s by x_s. The NBP algorithm alternates between sending messages to neighboring nodes, and computing marginals at nodes which received messages. We refer to each pair of steps, in which all messages are updated once, as an iteration.

To compute new outgoing messages, we use the marginal estimate $\hat{p}_s^{n-1}(x_s)$ from the previous iteration, which is represented by a set of samples. The inter-distance sensor model of Equation (5.2) implies that message samples are most easily generated in polar coordinates, with random orientation and radius perturbed from the noisy inter-distance reading:

$$\theta_{st}^{(i)} \sim \mathbb{U}(0, 2\pi) \qquad \nu_{st}^{(i)} \sim \mathbb{N}\left(0, \sigma_\nu^2\right) \tag{5.16}$$

$$x_{st}^{(i)} = x_s^{(i)} + (d_{st} + \nu_{st}^{(i)})[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})] \tag{5.17}$$

When the BP algorithm sends a message from s to t, the incoming message product (as in Equation (2.3)) avoids double-counting by excluding the message sent from t to s. To account for this in our NBP implementation, we use an importance sampling framework [183, 88]. Each sample $x_{st}^{(i)}$ is reweighted by $1/m_{ts}^{n-1}(x_s^{(i)})$, where $m_{ts}^{n-1}(x_s)$ is the preceding iteration's message (a Gaussian mixture). Also accounting for the effects of distance on the probability of receiving a measurement, the overall sample weight is

$$w_{st}^{(i)} = P_o(x_{st}^{(i)})/m_{ts}^{n-1}(x_s^{(i)}). \tag{5.18}$$

More generally, given a proposal density $q(x)$ from which we can draw samples $x^{(i)}$,

and a target density $p(x)$ we would like to approximate, importance sampling methods [49] assign a weight $w^{(i)} \propto p(x^{(i)})/q(x^{(i)})$ to each sample $x^{(i)}$.

Importance sampling methods are also used to approximate the marginal update of Equation (2.4). Recall that incoming messages $m_{ts}(x_s)$ are mixtures of M Gaussians. With $d = |\Gamma(s)|$ neighbors, the exact message product of Equation (2.4) will produce an intractable mixture with M^d components. To construct a proposal distribution, we instead take an equal number of samples $x_s^{(i)} \sim m_{ts}(x_s)$ from each incoming message. Combining these d groups of samples into a single set, they are then assigned importance weights

$$w_s^{(i)} = \frac{\prod_{t \in \Gamma(s)} m_{ts}(x_s^{(i)})}{\sum_{t \in \Gamma(s)} m_{ts}(x_s^{(i)})}. \tag{5.19}$$

To reduce importance sampling variance, Alg. 2 introduces a parameter $k > 1$. First, kM samples are drawn from the messages, and then M samples are drawn with replacement from the weights assigned to the kM samples.

Algorithm 1 *Phase 1, Compute Messages for Localization:* Given M samples $\{x_s^{(i)}\}$ representing marginal $\hat{p}_s^{n-1}(x_s)$, compute outgoing messages $m_{st}^n(x_t)$ for neighbors $t \in \Gamma(s)$

1. Draw $\theta_{st}^{(i)} \sim \mathbb{U}(0, 2\pi), \quad \nu_{st}^{(i)} \sim \mathbb{N}(0, \sigma_\nu^2)$
2. Means: $x_{st}^{(i)} = x_s^{(i)} + (d_{st} + \nu_{st}^{(i)})[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})]$
3. Weights: $w_{st}^{(i)} = P_o(x_{st}^{(i)})/m_{ts}^{n-1}(x_s^{(i)})$
4. Variance: $\Sigma_{st} = \sigma_\nu^2 \xi_M I$

Algorithm 2 *Phase 1, Compute Marginals for Localization:* Use incoming messages $m_{ts}^n = \{x_{ts}^{(i)}, w_{ts}^{(i)}, \Sigma_{ts}\}$ from neighbors $t \in \Gamma(s)$ to compute marginal $\hat{p}_s^n(x_s)$

1. **for** Neighbor $u \in \Gamma(s)$ **do**
2. Draw $\frac{kM}{|\Gamma(s)|}$ samples $\{x_s^{(i)}\}$ from each message m_{ts}^n
3. Weight each sample by $w_s^{(i)} = \prod_{u \in \Gamma(s)} m_{us}^n(x_s^{(i)}) / \sum_{u \in \Gamma(s)} m_{us}^n(x_s^{(i)})$
4. **end for**
5. Resample with replacement from these kM samples, producing M equal-weight samples.

5.3.1 Message Update Schedules

The convergence behavior of NBP depends on the order in which messages are transmitted between robotic nodes. In initial experiments, we found that a naive serial schedule, which iterates through each node one by one, performed poorly even with dozens of iterations and thousands of samples per message. Because sample-based density approximations cannot effectively populate large areas, biases introduced by poor initial message approximations can overwhelm informative messages sent by other robots. To resolve this, we devised an alternative schedule in which a node only transmits an outgoing message if it has received an incoming message from at least three neighbors. In situations where no node has three incoming messages, the threshold drops to two messages, and then finally to one.

5.4 Phase II: Tracking Algorithm

Phase I provides estimates of the locations of each robot at a single timestep. We then use phase II to incorporate the dynamics of the robot, allowing us to determine robot locations at subsequent timesteps more quickly, and improve accuracy by resolving ambiguities. Figure 5.3 provides an example with one robot that illustrates how the dynamics model can compensate for ambiguities.

In addition to sending messages between robots in the same timestep as the localization formulation does, our tracking formulation relates robots at the timestep τ with the same robot from the previous timestep $\tau - 1$, and the next timestep $\tau + 1$ when available. Given a sample of the node's current state $\chi_{s,\tau}^{(i)}$, the message sent to the next timestep is simulated from the dynamics model $\chi_{s,\tau+1}^{(i)} \sim p(\chi_{s,\tau+1} \mid \chi_{s,\tau}^{(i)})$. Because $\psi(x_{s,\tau+1}, x_{s,\tau}) = p(x_{s,\tau+1} \mid x_{s,\tau})$, when we send messages to a node at the previous timestep we fix $x_{s,\tau+1}^{(i)}$ and sample $x_{s,\tau}^{(i)} \sim \alpha p(x_{s,\tau+1}^{(i)} \mid x_{s,\tau})$, where α is a nor-

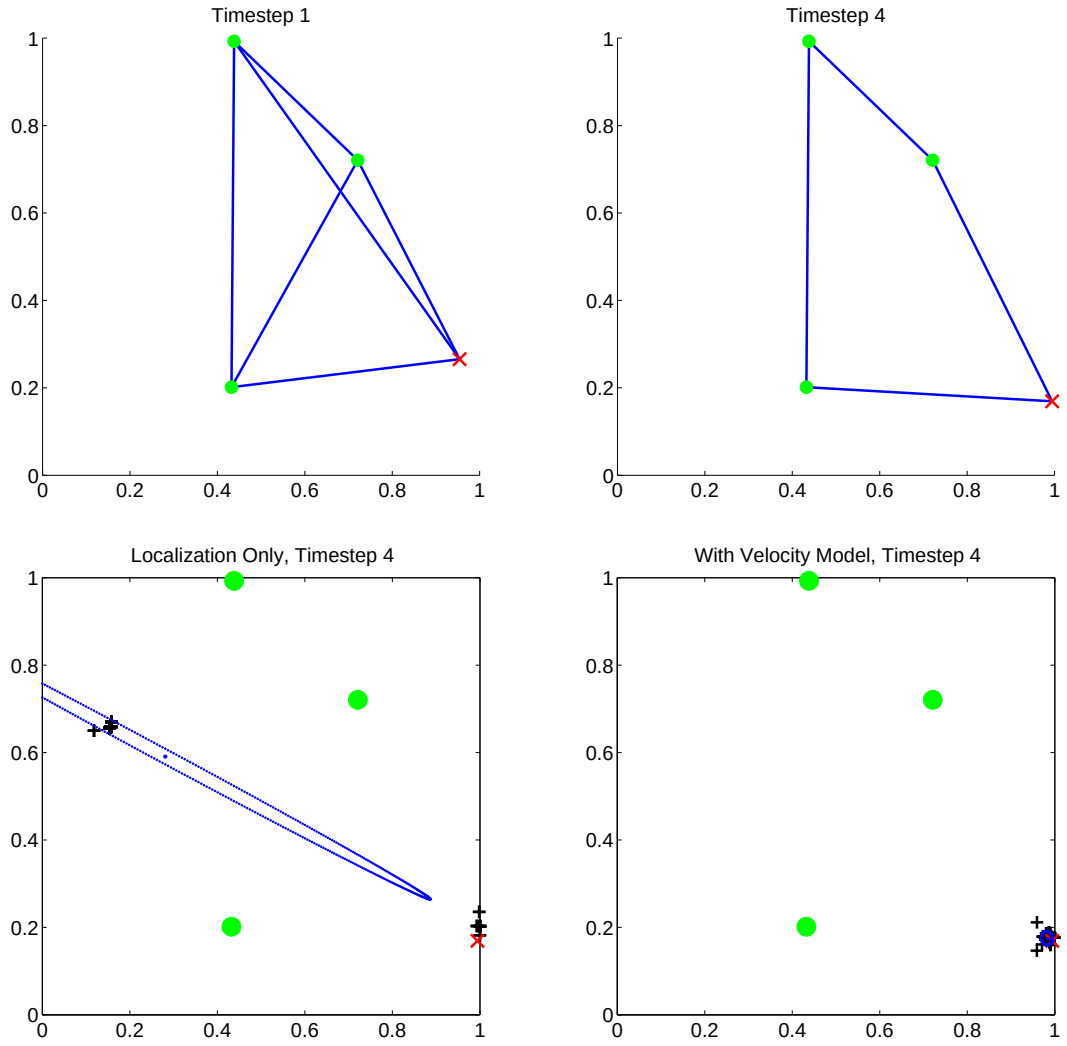


Figure 5.3. Illustration of dynamics model resolving bimodal position estimate for a single robot with three beacons. Using the same notation as in Figure 5.1, the upper figures show true locations of beacons and a robot in lower right of each plot. Samples from the robot marginal are indicated with black +s. Bottom figures show estimates of robot position for the 4th timestep, using a localization-only approach on the left, where there is substantial error, and using our tracking approach using dynamics on right, where error is negligible.

malization constant. Concretely, the update becomes

$$\begin{aligned} x_{s,\tau}^{(i)} &= x_{s,\tau+1}^{(i)} - \dot{x}_{s,\tau+1}^{(i)} \\ \dot{x}_{s,\tau}^{(i)} &= \dot{x}_{s,\tau+1}^{(i)} - \omega_{s,\tau} \qquad \omega_{s,\tau} \sim \mathbb{N}(0, \sigma_\omega^2) \end{aligned} \tag{5.20}$$

Our algorithm for computing marginals in phase II integrates temporal messages from the same robot at different timesteps, and spatial messages from neighbors at

the same timestep. We initially tested a simpler algorithm which propagates samples from the previous timestep according to the dynamics model, and then performs weighted resampling to account for new inter-distance readings. However, after more than four timesteps, this approach degraded to near-random estimates, even with thousands of samples M for each message. We hypothesize that this issue arises because the simpler formulation effectively only allows information to travel a single hop per time-step.

The more sophisticated approach of Algs. 3-4 overcomes these limitations by communicating multi-hop inter-distance information in a single timestep. Let $m_{s,\tau,\tau+1}(\chi_{s,\tau+1})$ denote the forward-time message sent by robotic node s at timestep τ, and $m_{s,\tau,\tau-1}(\chi_{s,\tau-1})$ the corresponding reverse-time message. As described in Section 5.2.3, we first decompose these messages into the marginal robot position distribution, and the conditional distribution of velocity given position; both are Gaussian mixtures. Using a generalized NBP message-passing algorithm, we then integrate the information provided by temporal dynamics and distance measurements at time τ. The resulting improved position estimates are then transferred to velocity estimates via the previously computed conditional densities. As summarized in the pseudocode, several stages of importance sampling with resampling are included to convert sets of message samples into the message products required by NBP, and ensure that samples (and hence computational resources) are efficiently utilized.

5.4.1 Message Update Schedules

Our tracking scheduler applies the phase II algorithms, after first initializing location estimates for each robot using phase I. Because we assume somewhat informative dynamics models, the uncertainty in robot locations at the next timestep is sufficiently peaked to allow a serial message update schedule. Our tracking framework is suffi-

Algorithm 3 *Phase 2, Compute Messages for Tracking:* Given M samples $\chi_{s,\tau}^{(i)} = \{x_{s,\tau}^{(i)}, \dot{x}_{s,\tau}^{(i)}\}$ from marginal $\hat{p}_{s,\tau}^{n-1}(\chi_{s,\tau})$, compute messages $m_{st,\tau}^{n}(x_{t,\tau})$ for neighbors $t \in \Gamma(s,\tau)$, $m_{s,\tau,\tau+1}(\chi_{s,\tau+1})$, and $m_{s,\tau,\tau-1}(\chi_{s,\tau-1})$

1. Draw $\theta_{st}^{(i)} \sim \mathbb{U}(0, 2\pi), \quad \nu_{st}^{(i)} \sim \mathbb{N}(0, \sigma_\nu^2)$
2. Means: $x_{st,\tau}^{(i)} = x_{s,\tau}^{(i)} + (d_{st,\tau} + \nu_{st}^{(i)})[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})]$
3. Weights: $w_{st,\tau}^{(i)} = P_o(x_{st,\tau}^{(i)}) / m_{ts,\tau}^{n-1}(x_{s,\tau}^{(i)})$
4. Variance: $\Sigma_{st,\tau} = \sigma_\nu^2 \xi_M I$
5. Means: $\chi_{s,\tau+1}^{(i)} \sim p(\chi_{s,\tau+1} \mid \chi_{s,\tau}^{(i)})$
6. Weights: $w_{s,\tau+1}^{(i)} = 1 / m_{s,\tau+1,\tau}^{n-1}(\chi_{s,\tau}^{(i)})$
7. Variance: $\Sigma_{s,\tau+1} = \mathrm{ROT}(\{\chi_{s,\tau+1}^{(i)}, w_{s,\tau+1}^{(i)}\})$
8. Means: $\chi_{s,\tau-1}^{(i)} \sim \alpha p(\chi_{s,\tau}^{(i)} \mid \chi_{s,\tau-1})$
9. Weights: $w_{s,\tau-1}^{(i)} = 1 / m_{s,\tau-1,\tau}^{n-1}(\chi_{s,\tau}^{(i)})$
10. Variance: $\Sigma_{s,\tau-1} = \mathrm{ROT}(\{\chi_{s,\tau-1}^{(i)}, w_{s,\tau-1}^{(i)}\})$

Algorithm 4 *Phase 2, Compute Marginals for Tracking* Use incoming messages $m_{ts,\tau}^{n} = \{x_{ts,\tau}^{(i)}, w_{ts,\tau}^{(i)}, \Sigma_{ts,\tau}\}$, $m_{s,\tau-1,\tau}^{n} = \{\chi_{s,\tau-1}^{(i)}, w_{s,\tau-1}^{(i)}, \Sigma_{s,\tau-1}\}$, and $m_{s,\tau+1,\tau}^{n} = \{\chi_{s,\tau+1}^{(i)}, w_{s,\tau+1}^{(i)}, \Sigma_{s,\tau+1}\}$ to compute marginal $\hat{p}_{s,\tau}^{n}(\chi_{s,\tau})$

1. Let $m_{s,\tau-1,\tau}^{n}(\chi_{s,\tau}) = m_{s,\tau-1}^{n}(x_{s,\tau}) m_{s,\tau-1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau})$
2. Let $m_{s,\tau+1,\tau}^{n}(\chi_{s,\tau}) = m_{s,\tau+1}^{n}(x_{s,\tau}) m_{s,\tau+1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau})$
3. Draw $\frac{kM}{3}$ samples $\{x_{s,\tau}^{(i)}\}$ from $m_{s,\tau-1}^{n}(x_{s,\tau})$
4. Draw $\frac{kM}{3}$ samples $\{x_{s,\tau}^{(i)}\}$ from $m_{s,\tau+1}^{n}(x_{s,\tau})$
5. **for** Neighbor $t \in \Gamma(s,\tau)$ **do**
6. Draw $\frac{kM}{3|\Gamma(s,\tau)|}$ samples $\{x_{s,\tau}^{(i)}\}$ from each $m_{ts,\tau}^{n}(x_{s,\tau})$
7. **end for**
8. Weight each of the kM samples by $w_{s,\tau}^{(i)} = \frac{m_{s,\tau-1}^{n}(x_{s,\tau}^{(i)}) \cdot m_{s,\tau+1}^{n}(x_{s,\tau}^{(i)}) \cdot \prod_{t \in \Gamma(s,\tau)} m_{ts}^{n}(x_{s,\tau}^{(i)})}{m_{s,\tau-1}^{n}(x_{s,\tau}^{(i)}) + m_{s,\tau+1}^{n}(x_{s,\tau}^{(i)}) + \sum_{t \in \Gamma(s,\tau)} m_{ts}^{n}(x_{s,\tau}^{(i)})}$
9. Resample with replacement from these kM samples, producing M equally weighted samples $\{x_{s,\tau}^{(i)}\}$.
10. Draw $\frac{kM}{2}$ samples $\dot{x}_{s,\tau}^{(i)} \sim m_{s,\tau-1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau}^{(i)})$, producing samples $\chi_{s,\tau}^{(i)} = \{x_{s,\tau}^{(i)}, \dot{x}_{s,\tau}^{(i)}\}$ from the joint marginal
11. Draw $\frac{kM}{2}$ samples $\dot{x}_{s,\tau}^{(i)} \sim m_{s,\tau+1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau}^{(i)})$, producing samples $\chi_{s,\tau}^{(i)} = \{x_{s,\tau}^{(i)}, \dot{x}_{s,\tau}^{(i)}\}$ from the joint marginal
12. Weight each of the kM joint samples by $w_{s,\tau}^{(i)} = \frac{m_{s,\tau-1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)}) \cdot m_{s,\tau+1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)})}{m_{s,\tau-1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)}) + m_{s,\tau+1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)})}$
13. Resample with replacement from these kM samples, producing M equally weighted samples $\{\chi_{s,\tau}^{(i)}\}$.

ciently general to perform smoothing, and use information from future timesteps to update location estimates for the current timestep. While alternative schedules are subjects for future work, we focus here in a filtering schedule which only propagates messages forward in time. We perform one tracking iteration following localization in timestep 1, and then a constant number of iterations at subsequent times.

5.4.2 Computational Complexity

The most computationally demanding portion of our tracker requires $\mathcal{O}(kM^2|E|)$ operations per timestep, where $|E|$ is the number of observed inter-distance measurements. The parameters k and M correspond to the number of samples used to represent distributions as described in the Nonparametric Belief Propagation subsection of Section 2.4.4. Our experiments use k = 3 and M varies from 100-500.

Most time is spent either sampling from Gaussian mixtures, or evaluating Gaussian mixtures at a particular location. Using a naive approach, both operations require $\mathcal{O}(M)$ operations for an M-component mixture. Binary search algorithms can be used to more efficiently draw samples in $\mathcal{O}(\log(M))$ time; our simulator implements this technique. We can improve the speed of Gaussian mixture evaluation by constraining estimated covariance matrices to be diagonal. Multiscale, KD-tree representations can also be used to accelerate evaluation and sampling for large M [87].

5.5 Experiments

We implemented a simulator to explore the performance of our algorithms. It is written in Java, and simulates all robots on a single core of a 2.2 GHz Pentium Core Duo. While we simulate all robots on a single machine, as our algorithms are distributed, we interpret the runtime per robot. Our experiments using $M = 100$

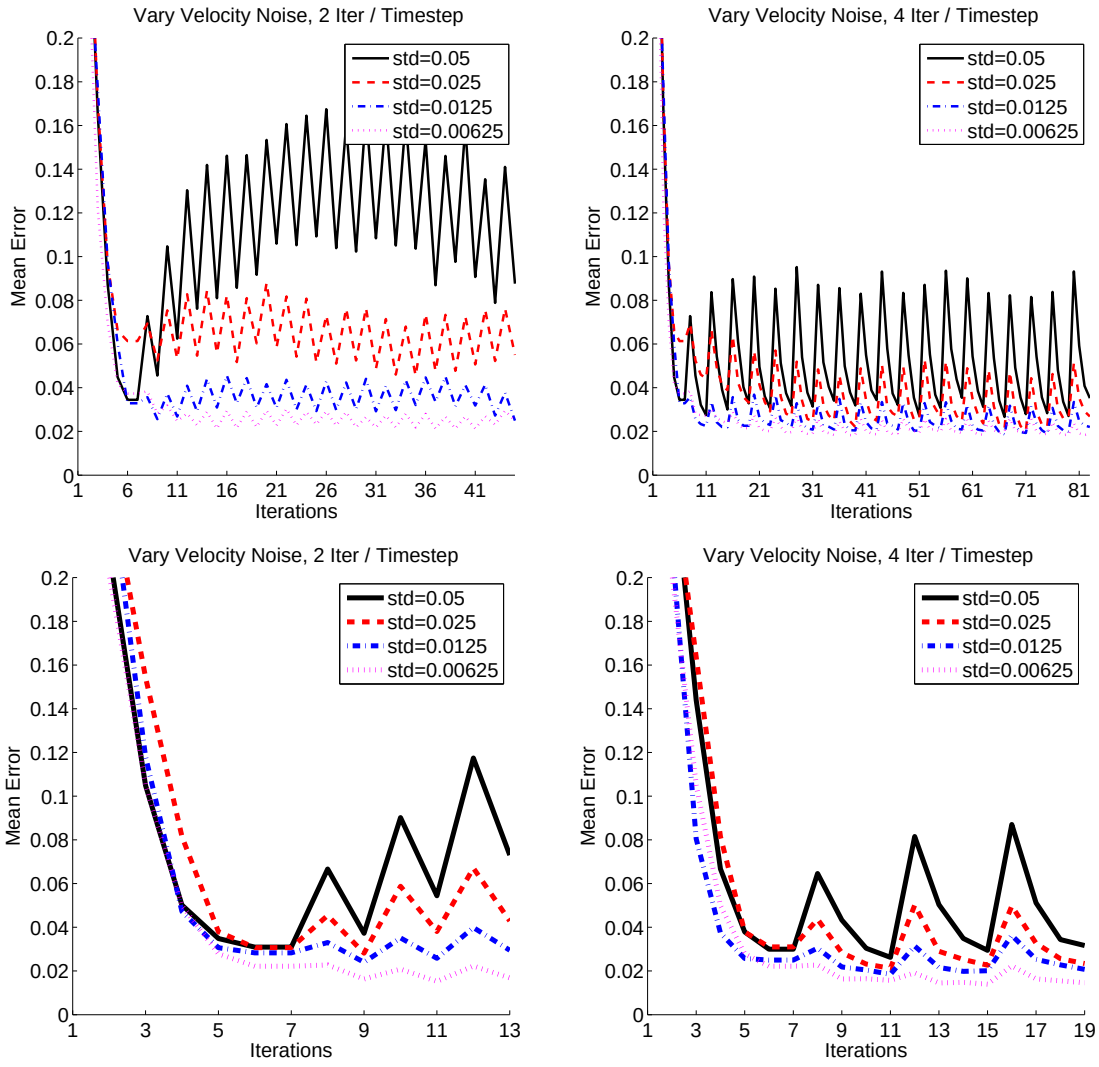


Figure 5.4. Tracking performance for datasets with varying levels of dynamics noise, using either two NBP iterations per timestep (left) or four iterations per timestep (right). We show performance over 20 timesteps on the top plots, and zoom in and show 4 timesteps in the bottom plots. The "sawtooth" shape of the mean error plot is due to a new timestep every two (in left plot) and four (in right plot) iterations. Using more iterations per timestep reduces localization errors, especially for noisier (less predictable) dynamics models, because information from further neighbors is incorporated (two hops in left, four hops in right). Our NBP algorithm makes use of multi-hop inter-distance readings, integrated over multiple iterations, to compensate for transition model errors.

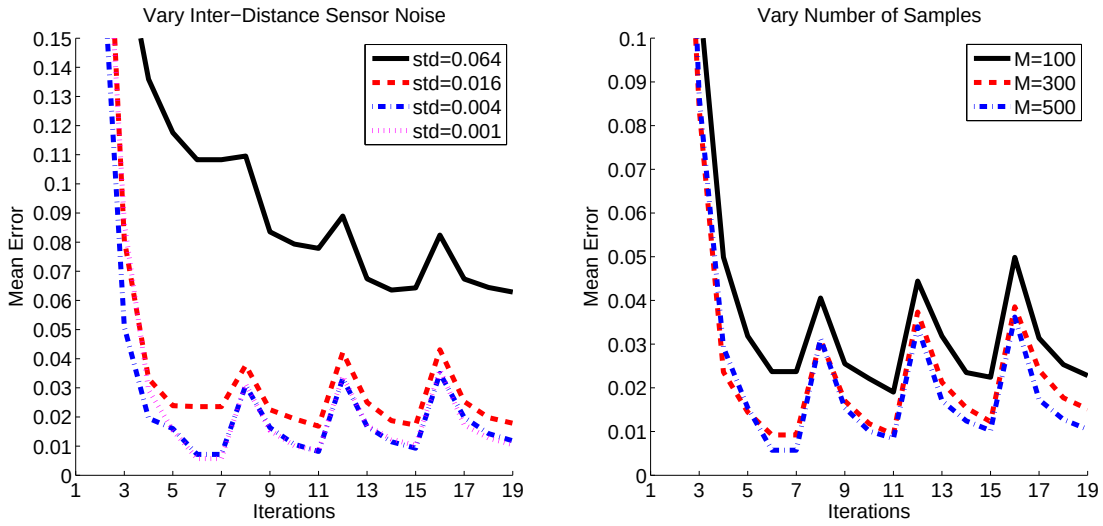


Figure 5.5. NBP tracking performance in various conditions. In the left plot, we use $M = 500$ samples, and vary the amount of inter-distance noise. In the right plot, we use very accurate inter-distance sensors ($\sigma_\nu = 10^{-3}$), and vary the number of samples used to represent each NBP message.

samples took approximately 0.65 seconds per robot per timestep. With $M = 500$ samples, this grows to 16.0 seconds per robot per timestep. As described in Sec 5.4.2, there are a number of remaining improvements which could be applied to improve effiency, and thereby allow real-time operation on mobile robots.

For each experiment, we use $n = 20$ robots and 3 stationary beacons, the minimum required for the tracking problem to be well posed. We generate the initial locations at $t = 0$ by randomly placing each robot in a 1x1 unit area. We then simulate each robot's movements according to the dynamics model $p(\chi_{s,\tau+1} \mid \chi_{s,\tau})$, producing locations at times $\tau = 2, \ldots, T$. Finally, we determine if robot s receives a distance reading from robot t according to $P_o(x_{s,\tau}, x_{t,\tau})$, and if so sample a noisy observation $d^{(i)}_{st,\tau} \sim p(d_{st,\tau} \mid x_{s,\tau}, x_{t,\tau})$.

As described in Section 5.4.1, we first apply phase I to localize robots at time $\tau = 1$, and then use phase II to track motion over time. Weighted marginal samples are used to produce estimates $\hat{x}_{s,\tau}$, which are used to evaluate our average estimation

error. We run each experiment 10 times, and plot the mean of our error metric across these trials. In Figure 5.1, we illustrate the connectivity from timesteps 1 and 4 for one trial, and the resulting discrepancy between the simulated (ground-truth) and inferred robot locations.

In Figure 5.4, we plot mean error over each iteration and vary the amount of noise in the dynamics model, allowing either two or four NBP iterations per timestep. The "sawtooth" shape of the mean error plot is due to a new timestep every two or four iterations, depending on the experiment. The error is worst at the first iteration of the new timestep, and then lessens as more iterations, providing more inter-distance information refine the estimate. With only two iterations per timestep, there is more error and variability than with four iterations per timestep. Extra iterations incorporate spatial information from robots further away, and compensate for unpredictable dynamics models. However, when the dynamics model is very accurate, and need fewer iterations per timestep to converge to accurate estimates. While localization of the first timestep takes six iterations, for more precise dynamics models, we only need two iterations of our tracking algorithm, resulting in a 3x savings in computation and communication.

In Figure 5.5, we vary the amount of noise in the inter-distance sensors, and test how estimation accuracy varies when many samples ($M = 500$) are available. Note that NBP's location estimates degrade gracefully as noise increases, with errors approximately proportional to the amount of sensor noise. The right panel of Figure 5.5 uses inter-distance sensors with low noise levels ($\sigma_\nu = 10^{-3}$), and examines how errors vary with the number of samples M. As desired, using additional samples does indeed lead to more accurate estimates.

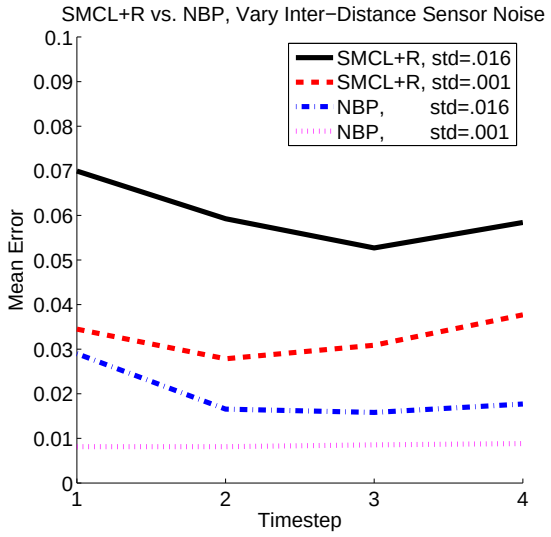


Figure 5.6. A quantitative comparison of the range-based SMCL (SMCL+R) formulation [46] to our NBP tracking algorithm. As our approach is more computationally intensive, we used $M = 5000$ samples for SMCL+R, and $M = 500$ samples for NBP. We present results with two inter-distance sensor noise levels. In both cases, NBP is 3-4 times more accurate.

5.6 Comparison with Range-Based SMCL

We compare our NBP tracker to the range-based, sequential Monte Carlo localization (SMCL+R) tracking formulation of Dil et al. [46]. In the SMCL+R approach, distances are approximated by the shortest path distance on the connectivity graph from each beacon to each mobile robot. Hop distances are computed from each beacon to each robot, and all beacon locations are communicated to each robot. To localize robots at each timestep, they use a polar model similar to ours. Samples are drawn at uniformly random angles around each beacon, but using a deterministic distance as determined by the shortest path to the robot of interest. A subset of these samples are then rejected, if they are either too far from the previous iteration's estimate (based on a maximum velocity model), or violate knowledge about the range of inter-distance sensors. More specifically, if a robot was a single-hop away, then samples should be within the maximum range, and if they were i hops away, the

samples must be within i times the maximum range. Samples are iteratively drawn at each timestep, for each robot, until M samples have not been rejected. For further implementation details, see [46].

As shown in Figure 5.6, the NBP tracker is 3-4 times more accurate than SMCL+R. However, the SMCL+R algorithm is faster, with a complexity of $\mathcal{O}(nb^2M)$ per timestep, if there are n robots and b beacons. We suspect our superior accuracy is in part due to our explicit modeling of sensor errors, rather than just using measured distances. We also avoid the approximations inherent in hop-based estimates of distances to beacons.

In some experiments, due to the hop-based distance approximation, and rejection of samples based on maximum transmission range, it was not possible to generate acceptable samples. As the approach assumes a unit-disk connectivity model [207], this was exacerbated by probabilistic connectivity and inter-distance measurement models. For example, with a probabilistic connectivity model P_o, if a robot is within the maximum transmission range from a beacon, but is two hops away, the correct samples for the robot will always be rejected. Thus, the experiment in Figure 5.6 used a unit-disk model for both approaches. By modeling probabilistic connectivity and inter-distance measurements, our approach is more robust as it does not experience to these problems.

5.7 Conclusion and Future Work

We present a new algorithm for collaborative robotic self-localization and tracking using nonparametric Belief Propagation (NBP). We compare the NBP dynamic tracking algorithm with SMCL+R, a sequential Monte Carlo algorithm [46]. Whereas NBP currently requires more computation, it converges in more cases and provides

estimates that are 3x to 4x more accurate. We are now working to reduce the computation time of the NBP algorithm.

The graphical modeling framework underlying our approach is very rich and we anticipate several extensions and generalizations. For example, we hope to explore how orientation estimates can improve localization accuracy. NBP can easily incorporate such information by appropriate modifications to the proposal distribution of Equation (9.3). Similarly, knowledge of environment geometry could be used to prune trajectory estimates that pass through obstacles. We are also working on "active" tracking where multi-modal estimates of robot position can be resolved by directing robots to move in directions that will reduce uncertainty.

Chapter 6

Security: SNARES

6.1 Introduction

Many new technologies for automated security, wireless networks, and sensing have emerged since the terrorist attacks of 9/11. Automation of security systems is an active area of research that also raises fundamental privacy concerns. In this chapter we consider the problem of automatically documenting an intruder's movements with a robotic camera, controlled by a sensor network's detections. We contrast this approach with the more conventional approach of using many static cameras, which would be significantly more expensive, and would require more space to store and effort to search through as there is significantly more footage.

New robotic cameras can be servoed to observe high-resolution detailed images of activity over a wide field of view. For example, the Panasonic KX-HCM280 pan-tilt-zoom camera (commercially available for under $1000.00) can capture up to 500 Mpixels per steradian. We are exploring how these cameras can be automatically controlled for security applications using a new class of inexpensive binary sensors with built-in wireless communications.

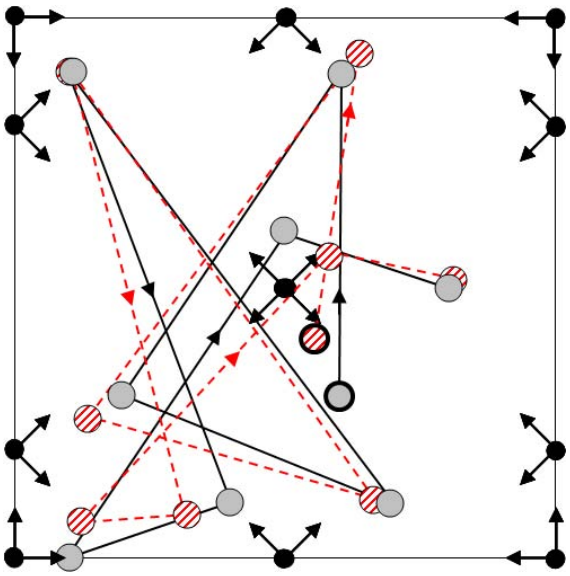


Figure 6.1. True vs. estimated intruder path for one randomized simulator run. Position and orientation of a network of fourteen motion sensors is indicated with solid dots and arrows. The true intruder path is indicated with hashed circles and dashed lines. The estimated intruder path is indicated with grey circles and solid lines. The bold-edged circles indicates the intruder's estimated and true starting points. Error is the spatial distance between true and estimated intruder position. In our simulation experiments, we report the distribution of error values.

Commercially available passive infrared (PIR) motion sensors (available for under $25.00 from x10.com) cover a field of view of 120 degrees. Wireless motion sensors like these are subject to two significant drawbacks: (1) they are binary and (2) after being triggered by motion in their field of view, they suffer from a refractory period of several seconds during which they are unresponsive. Since these and related sensors provide only coarse information about the presence or absence of an intruder, we propose a probabilistic tracking model based on Particle Filtering, where many sampled estimates ("particles") based on distributions of sensor response and intruder state are simultaneously estimated and fused to produce an aggregate point estimate of intruder position. Recent advances in computer processing make Particle Filtering feasible in real time. Using the estimate of the intruder's location the robotic camera actuates to document the intruder's movements for later analysis.

6.2 Problem Statement

In this section we formalize our problem. Our goal is to track (and document) the progress of an intruder moving through a known room using a sequence of binary sensor readings.

6.2.1 Inputs and Assumptions

Setup: polygonal room geometry, probabilistic sensor model, position and orientation of each sensor, probabilistic model of intruder movements, dimensions of intruder bounding prism. Input: time sequence of firing patterns from binary sensors. Output: estimate of intruder path.

Setup occurs once during initialization, while input is information that the system processes in real-time. Error is the spatial distance between estimated intruder position and actual intruder position. We report on the distribution of error values.

Room Geometry

The geometry of the room is described as a polygon represented by a set of vertices and edges. We denote all points in the room by the set R. We require that the room is star shaped so that a single robotic camera can be placed to have direct line of sight to any potential area the intruder may occupy. For now, we assume that there are no obstacles in the room.

Robotic Camera

Our system uses a single, robotic camera with a fixed base that focuses on specific areas within the observable space. The camera has three degrees of freedom: pan, tilt, and zoom.

Sensor Modeling

We have a set of $N = \{1 \dots n\}$ binary sensors, where each sensor i is modeled by two conditional probability density functions (CPDFs), each with a convex polygonal detection region R_i. Every point in the room must be covered by the region of at least one sensor:

$$R \subseteq \bigcup_{i=1}^{n} R_i$$

We use subset instead of equals because the union of the sensor region can exceed the room.

Each sensor i has a refractory period, whereby after triggering, it usually becomes unable to trigger again for another f_i seconds. The X10 PIR sensors we used for our experiments exhibit this problem.

We denote the generated sensor data by the set:

$$D = \left\{ (i,t) : \begin{array}{l} i \in N, t \in \mathbb{R}^+, \\ \\ \text{sensor } i \text{ triggers at time } t \end{array} \right\}$$

We process the sensor data generated from the intruder every t_P seconds, and thus we discretize time into iterations that are t_P seconds apart. Formally, the time t_τ of the τ'th iteration is defined as :

$$\forall \tau \in \mathbb{N} : t_\tau = \tau \cdot t_P$$

We use a world-coordinate system to integrate data across sensors. We overlay a uniform three-dimensional grid which discretizes the world into cells $j \in \{1 \ldots m\}$.

We define three indicator variables which are defined for each iteration. The event that an intruder fully occupied some world-space cell between the previous and current iteration:

$$O_{j,\tau} = \begin{cases} 1 & \text{if } \exists t : \text{an intruder fully occupies cell } j \text{ at time } t, t_{\tau-1} < t \leq t_\tau \\ 0 & \text{otherwise} \end{cases}$$

The event that the sensor is triggered between the previous and current iteration:

$$S_{i,\tau} = \begin{cases} 1 & \text{if } \exists t : (i,t) \in D, t_{\tau-1} < t \leq t_\tau \\ 0 & \text{otherwise} \end{cases}$$

The event that the sensor experiences a refractory period between the previous and the current iteration:

$$B_{i,\tau} = \begin{cases} 1 & \text{if } \exists t : (i,t) \in D, t_{\tau-1} - f_i < t \leq t_\tau \\ 0 & \text{otherwise} \end{cases}$$

Each cell in the grid corresponds to entries in two CPDFs. The first describe the probability that a sensor is not experiencing a refractory period and triggers, provided a sensor stimulus fully occupies only that cell:

$$P(S_{i,\tau} | O_{j,\tau} = 1, B_{j,\tau} = 0).$$

The second is the probability that a sensor is not experiencing a refractory period and triggers, provided no sensor stimulus occupies any part of that cell:

$$P(S_{i,\tau} | O_{j,\tau} = 0, B_{j,\tau} = 0).$$

Also, we need to compute the probability that a sensor fires, given that it is undergoing a refractory period:

$$P(S_{i,\tau} | B_{j,\tau} = 1)$$

With these three distributions, we completely describe for any iteration and sensor, the probability that a sensor fires.

Sensor Modeling Assumptions

For our particular sensor models, we make a number of important simplifying assumptions. We assume that the triggering of all sensors is independent given the intruder's state X_τ:

$$P(S_{1,\tau}, \ldots, S_{n,\tau}|X_\tau) = \prod_{i=1}^{n} P(S_i|X_\tau)$$

We assume that the probability that an intruder's complete occupancy of a single cell causes a sensor to trigger equals the probability that the intruder only occupies that cell:

$$P(S_{i,\tau}|O_{j,\tau} = 1, B_{i,\tau} = 0) =$$
$$P(S_{i,\tau}|O_{1,\tau} = 0, \ldots, O_{j,\tau} = 1, \ldots, O_{m,\tau} = 0, B_{i,\tau} = 0)$$

We assume that the probability of the occupancy of any cell causing the sensor to trigger is independent of any other cells causing the sensor to trigger:

$$P(S_{i,\tau}|O_{1,\tau}, \ldots, O_{m,\tau}, B_{i,\tau} = 0) =$$
$$\textstyle\prod_{j=1}^{m} P(S_{i,\tau}|O_{j,\tau}, B_{i,\tau} = 0)$$

We assume that during the sensor's refraction period, the distribution does not vary in the presence of any stimulus:

$$P(S_{i,\tau}|O_{1,\tau}, \ldots, O_{m,\tau}, B_{i,\tau} = 1) = P(S_{i,\tau}|B_{i,\tau} = 1)$$

We assume that the distributions of sensor's likelihood of firing given a sensor's position $P(S_{i,\tau}|O_{1,\tau}, \ldots, O_{m,\tau}, B_{i,\tau} = 0)$, smoothly changes between values over the detection region. When modeling a sensor type, we pick any sensor and view its parameters to be representative of all other sensors of the same type.

Intruder Modeling

We assume that we know the times the intruder enters and exits the room. We model our intruder by position, speed, orientation, and size. At each timestep, we add a sample from a zero-mean Gaussian to the intruder's speed and bound it by both the intruder's maximum speed v_{MAX} as well as the geometric constraints of the room. The orientation is also modeled by adding a sample from a zero-mean Gaussian, however the variance changes over time and is inversely proportional to the current speed of the intruder. We discuss this in more detail in the Intruder Model subsection of Section 6.6.1. To model size, we use a bounding rectangular prism, with given width, depth, and height. We model the path of the intruder as lines between samples, as is shown by the dashed line in Figure 6.1.

6.2.2 Outputs

The system uses the set of sensor firings to infer the intruder's path as illustrated in Figure 6.1. The objective is to minimize the error between the intruder's actual location x, and the system's estimation $\hat{x}$. It uses this information to generate a set of images represented by a unique tuple of pan, tilt, zoom, and time.

6.3 Framework

Our system involves three phases. In the Characterization Phase, we build a probabilistic model for the sensor. In the Deployment Phase, we place the sensors at the surveillance location and transform the distributions accordingly. Finally in the Tracking Phase, the system uses these sensor instantiations, in conjunction with firing information, to estimate the presence and location of an intruder traveling around

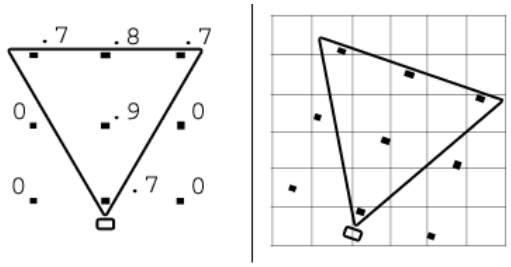


Figure 6.2. The left image depicts a two dimensional slice of a sensor's characterization, with the corresponding values at each point depicted in a grid-overlay. The right image shows a slice where the points have been transformed into worldspace.

a room. We use this information to direct a robotic camera to take photos of the intruder.

6.4 Characterization Phase

In this phase, we represent the properties of the sensor according to our sensor model. We begin by finding the duration of the sensor's refractory period f_i by continuously stimulating the sensor, and determine the rate at which it provides data. This sampling rate is determined by how often we will process the sensor data t_P. By comparing the differences between times the sensor transmits data of the stimulus, we compute f_i. In order to compute the probability that the sensor triggers while undergoing a refractory period, $P(S_{i,\tau} = 1 | B_{i,\tau} = 1)$, we stimulate the sensor and then monitor how often it signals an intruder for each iteration during its refractory period.

To calculate the CPDFs for each sensor-type, we perform a characterization according to a single sensor's coordinate space and overlay a three dimensional uniform grid over this space. By repeatedly stimulating the sensor restricted locally to the location of each index in the grid, we can calculate for each index the probability that a stimulus at only that location causes the sensor to fire. For index j, call this value

η_j. We depict a two-dimensional slice of sampling locations, which results in a grid of values Figure 6.2 (left). Each sample tests the sensor's response patterns to learn the empirical sensitivity of the sensor and how this sensitivity changes with respect to the location of the stimulus. In a PIR sensor, the stimulus might be a small amount of movement centered at the sample point. It is important to ensure that the samples are not taken during the sensor's refractory period and that the stimuli not overlap with the stimulus at another point on the grid. We describe in Section 6.5 how each sample point maps to an independent random variable, and avoiding stimulus overlap makes this independence assumption more reasonable.

6.5 Deployment Phase

Once we have a sensor type's characterization, we use a sensor's position, orientation and type to transform it into the shared world coordinate frame. A 2-dimensional layer of an example is shown in Figure 6.2 (right). We approximate the conversion by placing the center of the sample rotated from the sensor i's space into the containing cell j in world space; we use this as the value for $P(S_{i,\tau} = 1|O_{j,\tau} = 1, B_{i,\tau} = 0)$. To sufficiently characterize a sensor such that all m world-space cells have readings, we would need to sample all permutations of occupancies of cells, namely

$$P(S_{i,\tau}|O_{1,\tau} = o_{1,\tau}, \ldots, O_{m,\tau} = o_{m,\tau}, B_{i,\tau} = 0)$$

Acquiring all 2^m assignments is not feasible, so we make the assumptions described in Sensor Modeling Assumptions subsection of Section 6.2.1. We determine the values of all world-space cells within the convex hull of the cells containing non-zero η_js using Inverse Distance Weighting [174] and all other cells have a value of zero.

Because we do not have samples for all permutations for occupancies of cells, we also cannot calculate $P(S_{i,\tau}|O_{j,\tau} = 0, B_{i,\tau} = 0)$. This again leads us to the as-

sumptions described in the Sensor Modeling Assumptions subsection of Section 6.2.1 , which allow us to determine $P(S_{i,\tau}|O_{j,\tau} = 0, B_{i,\tau} = 0)$ by using the frequency of sensor false positives. Let this rate be h, the number of world-space cells where $P(S_{i,\tau}|O_{j,\tau} = 1, B_{i,\tau} = 0) > 0$ be ξ, and these cells be independent. Then

$$P(S_{i,\tau} = 1|O_{j,\tau} = 0, B_{i,\tau} = 0) = 1 - (1 - h)^{1/\xi}$$

Taking the convex hull of cells with non-zero values mapped to them yields the sensor shape's convexity requirement.

6.6 Tracking Phase

A state is a set of information that describes the relevant properties of an object that evolves over time, such as position and velocity. The probabilistic model of how states transition over time is referred to as the transition model. Because our state estimation is tracking of the intruder, we refer to this as our intruder model. A probabilistic model of the observable evidence, given some unobservable state, is the observation or sensor model. If the system can be described by these two models, it can be represented as a Dynamic Bayesian Network (DBN) [162]. If the intruder model and sensor model are linear functions of Gaussians, where the new state is a linear function of the previous state plus Gaussian noise, then a Kalman Filter can be used to provide the optimal solution to the state estimation problem.

While our intruder model, which is presented in Section 6.6.1 is a linear function of Gaussians, our sensor model is not. Because our sensor model is only restricted to a convex-shaped CPDF, there are two options: an Extended Kalman Filter or a Particle Filter. We can use a function to estimate our sensor model, for instance a spline, and then we can use an Extended Kalman Filter. Instead, we choose Particle Filtering because it can be used directly by sampling the CPDFs.

6.6.1 Tracking with Particle Filtering

To use Particle Filtering, we need to define the state of our system at every iteration $x_{i,\tau}$ and the observed evidence z_τ. By using previous phases of the system, we derive the distributions for $P(X_0), P(X_\tau|X_{\tau-1})$, and $P(Z_\tau|X_\tau)$ using Particle Filtering.

Intruder Model

The state of our system for each sample is represented by a 5 tuple of x-position, y-position, orientation, speed, and if the sensor is experiencing a refractory period.

While the rest of our system uses three-dimensions, the intruder model does not change its location along the z-axis. This is because the placement of the sensors and the camera will not necessarily be in the same plane, but it is reasonable to assume the intruder will remain on the floor.

Because we are using Particle Filtering, the intruder model is a function that maps from the old state and yields some new state. The transition model we chose $P(X_\tau|X_{\tau-1})$ determines position by Euler Integration. The standard deviation of the Gaussian added to current speed is determined according to typical properties of our intruder. The Gaussian added to the orientation has a standard deviation that is inversely proportional to the current speed because the faster our intruder is traveling, the more likely it is that the intruder's path will not deviate significantly from the current one.

Define $\beta_{i,\tau}$ as deterministic variables, timestamped every time their respective $S_{i,\tau}$ is true. If there sensor is triggering, we set β_i, τ to be the blind time. Otherwise, the $\beta_{i,\tau}$ for sensor i subtracts off t_P from the previous timestep, and is restricted to be at

least 0. Formally:

$$\beta_{i,\tau} = \begin{cases} f_i & \text{if } S_{i,\tau} = 1 \\ \max(0, \beta_{i,\tau-1} - t_P) & \text{else} \end{cases}$$

The random variable $B_{i,\tau} = 1$ iff $\beta_{i,\tau} = 0$.

We define $P(X_0)$ for our system as uniform at random for position, orientation and speed. This is sufficient because the system converges fast enough to the right distribution that no pre-seeding is necessary. Whenever the system begins, it sets $\beta_{i,0} = 0, \forall i \in N$ so that all sensors are defined as not undergoing a refractory period.

As the transitions discussed above have no knowledge of the room geometry, we need to impose the room's constraints on the transition model. To add these constraints, we use rejection sampling, first presented in [200]. We propose an update to our state according to our transition model. If the position is within the room and the speed is between 0 and v_{MAX}, then we accept this sample. If our proposal does not fit within these bounds, we continue to try new updates to the old state until we find one that satisfies these constraints.

Sensor Model

Denote the set of evidence among all sensors by:

$$z_\tau = \{S_{1,\tau} = s_{1,t}, \ldots, S_{N,\tau} = s_{N,\tau}\}$$

Because any sensor triggering is independent of all others given the location of the intruder:

$$\begin{aligned} P(Z_\tau|X_\tau) &= P(S_{1,\tau}, \ldots, S_{N,\tau}|X_\tau) \\ &= \textstyle\prod_{i=1}^{N} P[S_{i,\tau}|X_\tau] \end{aligned}$$

In order to compute the probability that a specific sensor i fires at iteration τ, due to an intruder of state x we use two different distributions. If the sensor is undergoing

a refractory period, we use:

$$\begin{aligned} & P(S_{i,\tau} = 1 | X_\tau = x_\tau) \\ & = P(S_{i,\tau} = 1 | B_{i,\tau} = 1) \end{aligned}$$

Otherwise, we use:

$$\begin{aligned} & P(S_{i,\tau} = 1 | X_\tau = x_\tau) \\ & = 1 - \prod_{k=1}^{m} \mathrm{L} \begin{pmatrix} P(S_{i,\tau} = 0 | O_{k,\tau} = 0, B_{i,\tau} = 0), \\ P(S_{i,\tau} = 0 | O_{k,\tau} = 1, B_{i,\tau} = 0), \\ \mathrm{V}(O_{k,\tau}, x_\tau) \end{pmatrix} \end{aligned}$$

We use L(a, b, c) to mean the linear interpolation of percentage c between a and b. V(a, b) is the volume percentage of cell a occupied by intruder with state b for a given bounding prism size.

6.6.2 Estimator

The Particle Filter gives us a mechanism to determine our confidence that an intruder is in a specific region. After the resampling step, the samples of the state space have been distributed according to the likelihood of the state being correct. The higher the density of particles in a region, the more likely that area is occupied by the intruder. Because the intruder's dimensions are approximated by a prism that bounds the size of the intruder, to determine the intruder's location at each timestep τ, we iterate through each world-space cell and count the number of samples of our state that reside within the bounding prism centered at the cell's position. We choose the cell that maximizes the number of samples as the estimated location of the intruder.

Once we estimate the location of the object in world space, we calculate the pan, tilt, and zoom necessary for the camera to take a photo. Because we bound the intruder with a rectangular prism, we pick the parameters that make the camera view the entire bounding prism, but no more. To determine pan and tilt, we find

the center of the intruder bounding box, and then convert from cartesian to spherical coordinates, where the origin in spherical coordinates is the location of the camera. Next, we project the intruder bounding prism onto an image plane orthogonal to the viewing plane and determine the correct zoom to view just the projection, but no more. There is a direct mapping between the location of the bounding prism in world-space and a specific pan, tilt, and zoom for the camera that maps to this location.

6.7 Simulation Results

We implemented a simulator to evaluate our estimator. First, it generates random intruder paths and probabilistic sensor models to compute corresponding sets of sensor data. Second, it processes this data to generate estimates of the intruder's location. We compare the true positions with the estimated intruder positions over time and compute error based on Euclidean distance. The simulator runs on a 1.6 Gigahertz Pentium M with 768 Megabytes of RAM using 1000 state samples (particles).

We performed simulation experiments for three sensor types: (1) perfect optical beams, (2) perfect motion sensors, and (3) imperfect motion sensors. For each, we considered refractory periods of length 0, 4, 8, and 16 seconds. We also consider the effect of varying the number of binary sensors in the network.

In all simulations, we use a square room of size 10 x 10 x 10 cells of unit size. We defined our intruder bounding prism of 1 cell height and 2 cell width and length. We use the parameters: $t_P = 2$ seconds, $v_{MAX} = 4$ cells per second and velocity Gaussian standard deviation is 1.5 cells per second, and $P(S = 1|B = 1) = 10^{-6}$. We generated 10 3-minute trajectories evaluated at every 2 seconds according to this model, which we then use across all tests. The locations of the intruder at each iteration x_τ is

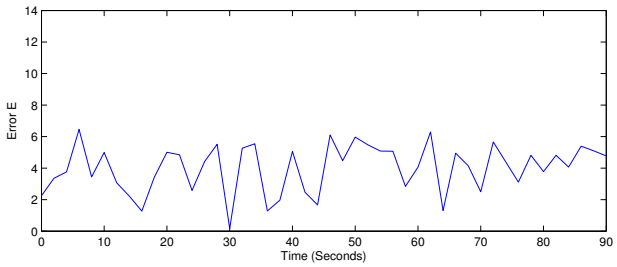


Figure 6.3. Error in estimated intruder position over time for the baseline Naive Estimator that simply reports the intruder is stationary in the middle of the room.

used to determine which sensors trigger. We send this set to the server, which then responds with its estimate $\hat{x}_\tau$. We determine our error E by the Euclidian distance between the estimation and the true state $|x_\tau - \hat{x}_\tau|$.

6.7.1 Baseline Naive Estimator

As a baseline for comparison, we consider the naive estimator that simply reports that an intruder is stationary in the center of the room. We compute the error values for this estimator on a simulated intruder path and compare them with those from our estimator.

First, we plot the point error values over time in Figure 6.3. Over the entire path, the Baseline Naive Estimator has an average error of 2.9 cells. The error is fairly well dispersed over the domain, with a slight trend of high error staying high, and low error staying low. This is because if the intruder is in a corner at one timestep, it is likely to remain near the corner in the next timestep. Next we present the distribution of error over all runs in Figure 6.4 and see that 55% of particles are less than 4 cells away.

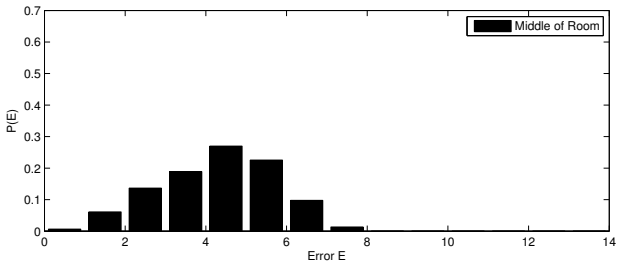


Figure 6.4. Baseline Naive Estimator: Distribution of error values.

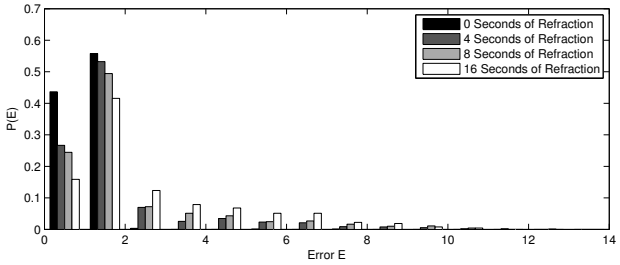


Figure 6.5. Twenty-two Perfect Optical-Beam Sensors: Distribution of error values.

6.7.2 Twenty-Two Perfect Optical Beam Sensors

Next we consider a set of binary optical beam sensors. These sensors have perfect detection, with no false positives or negatives. Their region of detection is 10 x 1 x 1 cells. We place these sensors at height (z value) 0, with one set of sensors going along the floor in the x direction for each cell index of y. We do the same in the y direction for each cell index in x, yielding a criss-crossing pattern of these sensors. Thus, if exactly one in the x set, and one in the y set trigger, we know within 1 world-space cell where the intruder is. We placed the intruder at height 0 and found the distributions illustrated in Figure 6.5.

This test illustrates a sanity check for a 0 second refractory period, as with perfect sensors, all estimations should be within two world-space cells. We also can see that we degrade gracefully as we increase our refractory period, and even with a 16 second refractory period, we still get 64% within two cells.

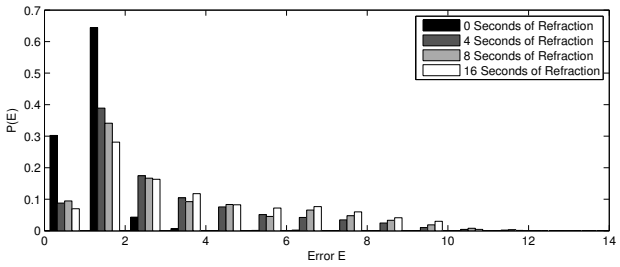


Figure 6.6. Fourteen Perfect Motion Sensors: Distribution of error values.

6.7.3 Fourteen Perfect Motion Sensors

We then performed a setup with 14 perfect motion sensors sensors with a characterization of η where for cells of height 0, $\eta_i = 1$ in the form of a isosceles triangle, with a resolution of a reading every two world-space cells. For all other heights, $\eta_j = 0$.

We present the sensor placement, orientation and an example run, with values shown for every 10 seconds for a perfect motion sensor with no refraction in Figure 6.1. This configuration generates a large number of intersections, allowing for higher quality localization. We present the distributions of error in Figure 6.6.

As we would expect due to the lower number of sensors, with more complicated form of intersection, this setup produces poorer quality localization of the intruder, even with no refractory period. It was interesting to note that there is little difference in localization between 4 and 8 second refractory periods. This is because there is sufficient overlap to compensate for the sensors experiencing refractory periods. However, the compensation is less robust in this simulation, as the change from a 0 to 4 second refractory period was much more significant.

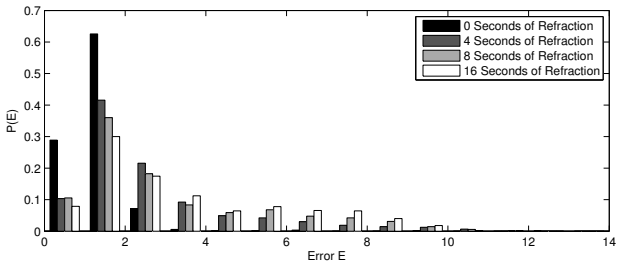


Figure 6.7. Fourteen Imperfect Motion Sensors: Distribution of error values.

6.7.4 Fourteen Imperfect Motion Sensors

Next, we modeled a sensor which becomes less sensitive as we stimulate it further away. We then ran the same tests with the new characterization that had false negatives which increased as a function of distance from the sensor, and a false negative rate $P(S_{i,\tau} = 1 | O_{1,\tau} = 0, \ldots, O_{m,\tau} = 0, B_{i,\tau} = 0) = 0.25$.

These distributions are very close to those observed with the perfect sensor over all refractory periods, though as expected the perfect sensors perform slightly better.

6.7.5 Varying Number of Sensors

By varying the number of sensors, it becomes evident that increasing the number of sensors, and thereby allowing more overlap for sensors to compensate one another, helps accommodate for the refractory period. While 4 Sensors does not do much better than our baselines, 8 sensors does better, and 14 does significantly better. We present this in Figure 6.8.

6.7.6 Errors over Time for Imperfect Motion Sensors

We display the error in estimated intruder position over time for the imperfect motion sensors with a 4 second refractory period in Figure 6.9. Note that there is

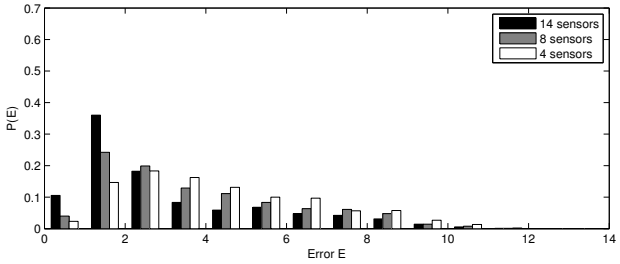


Figure 6.8. Varying Number of Binary Motion Sensors: Distribution of error values for different number of binary motion sensors (each with 8 second refractory period).

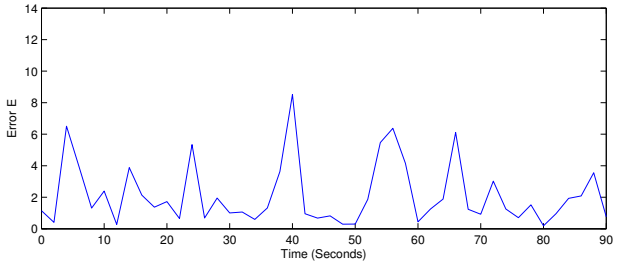


Figure 6.9. Error in estimated intruder position over time for Fourteen Imperfect Motion Sensors (each with 4 second refractory period).

high variance in the error values. High values result when all sensors are blind in the region where the intruder is. However, the estimator quickly compensates, and is able to recover.

6.8 In-Lab Experiment

For the in-lab experiment, we ran tests on 8 X10 PIR sensors, which have an 8 second refraction time. The lab is 8 x 6 meters, and we use world-space cells of size .3 meters. Here, we present a sample of photos from our system. We used the parameters: $t_P = 2$ seconds, $v_{MAX} = 4$ feet per second and velocity Gaussian standard deviation is 1.5 feet per second. Again, we used 1000 samples of the distributions. In our test shown in Figure 6.10, we had an intruder walk around the room, and

compare photos of the estimated path with the real path. This example shows that our system performs well in securing the lab.

6.9 Conclusion and Future Work

We have described a system that uses a network of binary motion sensors to track an intruder. We begin by defining probabilistic sensor models that include refractory periods. We develop a new method for processing noisy sensor data based on Particle Filtering that incorporates a model of intruder velocity. We report experiments with this method using a new simulator and using physical experiments with X10 sensors and a robotic pan-tilt-zoom camera in our laboratory.

In the future, we will experiment with different sensor models and different spatial arrangements of sensors, and set up the camera system to run over extended duration in our lab. An interesting open problem optimal sensors placement, which can be considered a variant of the Art Gallery problem. We will also investigate methods that can simultaneously track multiple intruders. We are also interested in ways to decentralize the algorithm by moving processing onto a network of smart sensors. We would like to investigate how altering parameters, such as the number of samples, or data processing frequency affects performance. Lastly, we would like to use vision processing techniques to utilize information gathered from the camera to enhance our system.

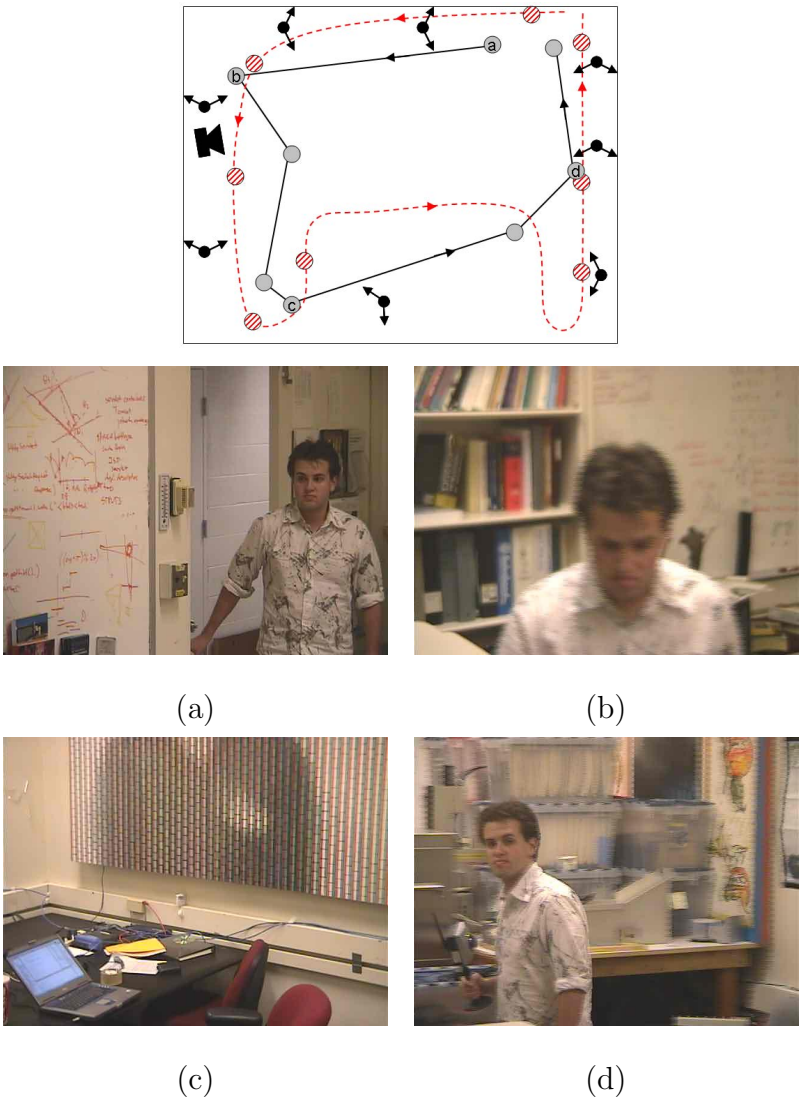


Figure 6.10. Lab Experiment: A map of our lab (top), with intruder true path indicated by a dashed line as in Fig 6.1. Estimated path indicated by a solid line. Photos taken by a robotic camera (a-d) correspond to the numbered solid blue dots and their respective locations in the room.

Chapter 7

Security: Actuator Networks

7.1 Introduction

As robots become smaller and simpler and are deployed in increasingly inaccessible environments, we need techniques for accurately guiding robots in the absence of localization by external observation. We explore the problem of observation- and localization-free guidance in the context of *actuator networks*, distributed networks of active beacons that impose guiding forces on an unobserved robot. Because our approach does not have the robot directly communicate with beacons, the approach can also be used to guide a tiny robot for covert surveillance.

In contrast to sensor networks, which passively observe their environment, actuator networks actively induce a physical effect that influences the movement of mobile elements within their environment. Examples include light beacons guiding a robot with an omni-directional camera through the dark, electric fields moving a charged nano-robot through a fluid within a biological system, and radio transmitters on sensor motes guiding a robot with a single receiver. We consider a network of beacons which exert a repellent force on a moving element. This repellent effect models

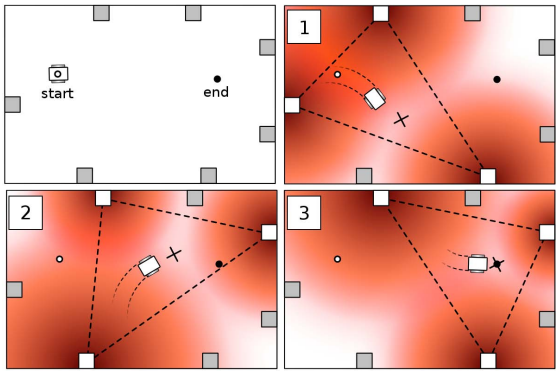


Figure 7.1. An actuator network with sequentially activated actuators triplets (shown as squares) driving a mobile robot toward an end location. The robot is guided by creating locally convex potential fields with minima at waypoints (marked by ×s).

many systems for which purely attractive models for beacon-assisted navigation are not realistic or practical. The repulsive effect of an actuator network permits the guided element to pass through specific points in the interior of a region while avoiding contact with the actuators themselves, unlike traditional attractive models for beacon-assisted navigation. Due to their simple structure, actuators can be low-cost, wireless, and in certain applications low-power.

In this chapter, we consider a specific application of actuator networks: guiding a simple mobile robot with limited sensing ability, unreliable motion, and no localization capabilities. A motivating scenario for such a system emerges from the potential use of low-cost robots in hazardous-waste monitoring and cleanup applications, such as the interior of a waste processing machine, nuclear reactor, or linear accelerator. Robots in these applications must execute cleanup and monitoring operations in dangerous, contaminated areas where, for safety or practical reasons, it is impossible to place human observers or cameras, or to precisely place traditional navigational waypoints. In this scenario, an actuator network of radio or light beacons may be semi-randomly scattered throughout the region to aid in directing the robot from location to location through the region. By shifting responsibility for navigation to a network of actuators

which directs a passive robot, smaller and more cost-effective robots may be used in these applications.

We consider the 2D case, where the actuator network consists of beacons at different locations in a planar environment. The actuator network is managed by control software that does not know the position of the robot, but it is aware of the positions of all of the actuators. Such software may be either external to the system, or distributed across the actuators themselves, as in the case of a sensor network. We assume only that the robot's sensor is able to detect and respond consistently to the strength and direction of each actuation signal. Such a general framework accurately represents a wide variety of real-world systems in use today involving deployed navigational beacons, while significantly reducing the technical requirements for both the beacons and the robots.

We present an algorithm that sequentially switches between sets of actuators to guide a robot through a planar workspace towards an end location as shown in Figure 7.1. Using three non-collinear actuators, each one generating a repellent force field with intensity falling off inversely proportional to the square of the distance, we generate potential fields that drive the robot from any position within the convex hull of the actuators to a specific position within the interior of the triangle formed by the actuators (the local minimum of the potential field). The optimal sequence of potential fields to move the robot to a specific point within the interior of the workspace is determined using a roadmap that incorporates the possible potential fields as well as uncertainty in the transition model of the robot. Despite this uncertainty and the absence of position measurement, the locally convex nature of potential fields ensures that the robot stays on track. We present results from simulated actuator networks of small numbers of beacons in which a robot is successfully guided between randomly chosen locations under varying models of motion uncertainty. With as few as 10 actuators in a randomly-placed actuator network, our algorithm is able to steer a robot

between two randomly selected positions with probability 93.4% under motion uncertainty of 0.2% standard deviation Gaussian Polar motion uncertainty (perturbing the robot's magnitude and angle with additive Gaussian motion uncertainty).

7.2 Problem Formulation

7.2.1 Assumptions

We consider the control of a single mobile robot in a planar environment, influenced by an actuator network. The n actuators are located at known positions $\mathbf{x_i} \in \mathbb{R}^2$. Each actuator can be controlled independently to produce a signal with piece-wise constant amplitude $a_i(t) \geq 0$. Each actuator generates a radially symmetric potential field U_i of the form

$$U_i(\mathbf{x}) = \frac{a_i}{|\mathbf{x} - \mathbf{x}_i|} \tag{7.1}$$

The direction and magnitude of the gradient of this field can be observed from any location $\mathbf{x} \in \mathbb{R}^2$ and are given by the vector field $\mathbf{F_i}$

$$\mathbf{F_i}(\mathbf{x}) = \frac{\partial U_i}{\partial \mathbf{x}} = -\frac{a_i(\mathbf{x} - \mathbf{x_i})}{|\mathbf{x} - \mathbf{x_i}|^3} \tag{7.2}$$

i.e. the signal strength $|\mathbf{F_i}(\mathbf{x})|$ is inversely proportional to the square of the distance to the actuator, as is common for physical signals.

The aim of the actuator network is to guide a mobile robot along the direction of steepest descent of the combined potential field $U(\mathbf{x}) = \sum_i U_i(\mathbf{x})$. The position of the robot as a function of time is denoted by $\mathbf{p}(t) \in \mathbb{R}^2$, and the robot is assumed to have sufficient local control to be able to move approximately in a given direction vector $\mathbf{v}$ (relative to its own coordinate frame). The desired motion direction $\mathbf{v}$ is equal to the direction of steepest descent of the combined potential field U and is

hence given by the vector sum

$$\mathbf{v} = -\sum_i \mathbf{F_i}(\mathbf{x}) = \sum_i \frac{a_i(\mathbf{x} - \mathbf{x_i})}{|\mathbf{x} - \mathbf{x_i}|^3} \tag{7.3}$$

Under this control strategy, each of the actuators serves effectively as a "robot repulser", causing the robot's direction of motion to be determined by the total combined force from all actuators. We only assume that the robot can continuously measure the direction and strength of the superimposed actuator signals. In particular, no global position sensors or odometry is required. Its motion is described by a general model $\dot{\mathbf{p}} = R(\mathbf{v})$, which may include stochastic components to represent sensor and actuator uncertainty, or general uncertainty in movement across the workspace due to viscosity in a fluid environment, uneven terrain causing wheel slip, or simply unreliability in the robot's design.

The design of a suitable sensor for the robot to measure the actuator signals depends on the type of actuation used. For light-based actuation, one could use an omni-directional camera. Each actuator signal from a certain direction causes a specific spot in the image to light up with a brightness depending on the actuator strength and distance. The desired motion vector $\mathbf{v}$ can then be computed simply by adding the force directions corresponding to all image pixels, weighted by their brightness.

A centralized controller is assumed to know the location of the actuators and to be able to control the actuator amplitudes in a time-discrete manner. It does not have other sensing information; in particular, it cannot measure the robot's position.

7.2.2 Inputs and Output

The inputs of the control algorithm are the initial location $\mathbf{p_0} = \mathbf{p}(0) \in \mathbb{R}^2$, the end location $\mathbf{p_e} \in \mathbb{R}^2$ of the robot, and the locations of each actuator $\mathbf{x_i} \in \mathbb{R}^2$. The

actuator locations can either be determined a priori, or via some localization scheme. The key aspect is that the actuators will not observe or track the robot.

The proposed algorithm returns a sequence of actuator amplitudes $\{a_i(t_j)\}$ at discrete time instants t_j that maximizes the probability that the robot successfully moves from the start location $\mathbf{p_0}$ to the end location $\mathbf{p_e}$. If no path from $\mathbf{p_0}$ to $\mathbf{p_e}$ can be found, the algorithm returns the empty sequence. Each set of amplitudes contains exactly three nonzero values corresponding to a particular triangle of actuators and an associated start and destination location. The use of only three actuators at a time has the advantage of simplifying the analysis of the system and potentially reducing power consumption of the network by limiting the number of active actuators at any time. To conserve power, all idle actuators can go into a low-power state. Each time instance is separated by a sufficiently large duration that a robot starting at the associated start location will by this time either have migrated to the associated target location moving at minimum velocity or will be outside the actuator triangle and get progressively farther away. This time is determined by the minimum speed of the robot.

7.2.3 Motion Uncertainty

To investigate the utility of actuator networks in steering robots with *uncertain* motion, we considered two uncertainty models $R(v)$: one using additive random Gaussian noise on the robot's Cartesian coordinates, and one using additive random Gaussian noise on the robot's polar coordinates. In other words, the Cartesian motion uncertainty model describes the uncertain motion of the robot as

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \dot{\mathbf{p}} = R(\mathbf{v}) = \begin{bmatrix} N\left(v_x, \sigma^2\right) \\ N\left(v_y, \sigma^2\right) \end{bmatrix} \tag{7.4}$$

while the polar motion uncertainty model is given by

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \dot{\mathbf{p}} = R(\mathbf{v}) = \begin{bmatrix} r\cos(\theta) \\ r\sin(\theta) \end{bmatrix} \tag{7.5}$$

where r and θ are drawn from Gaussian distributions as

$$r = N\left(|\mathbf{v}|, \sigma^2\right) \qquad\qquad \theta = N\left(\text{atan2}(v_y, v_x), \frac{\sigma^2}{4\pi^2}\right)$$

7.3 Motion control using actuator networks

To solve the problem of finding a valid actuation sequence (if one exists), the algorithm first generates all possible $\binom{n}{3}$ triangles that can be formed using the n actuators. Then, it computes the incenters of these triangles (as discussed in Section 7.3.1) to be used as local minima of the potential fields. These incenters define the vertices in a graph, and the next step of the algorithm is to determine the weights of all possible edges between the vertices. We define the weight of an edge to be the probability that the robot successfully navigates from one vertex to the other, as defined by the robot motion model $R(\mathbf{v})$. The resulting graph defines a roadmap for the robot, and the last step of the algorithm is to insert the start and end locations into the roadmap and determine the path between them that maximizes the probability of successfully reaching the end location.

7.3.1 Local control using actuators triplets

The following assumes the actuators we choose are not collinear. Consider a potential field U generated by an actuators triplet i, j, and k (and all the other actuators in the network set to zero amplitude). If the amplitudes of the actuator triplet is strictly positive, the potential field will have a local minimum at some point

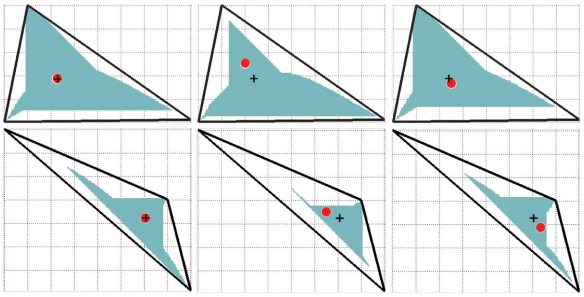


Figure 7.2. Examples of two different triangles and three different waypoints (denoted as circles) and their capture regions (shaded areas) for both triangles. The incenter of each triangle is marked with a '+'.

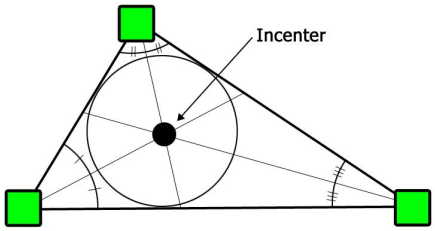


Figure 7.3. Example of a triangle with its incenter.

inside the triangle. This location is called the *waypoint* of the potential field. In our global algorithm, we may traverse many waypoints to get from the start to the end location. The final waypoint is defined to be the end location. For a given triangle structure, we define the *feasible region* as the set of locations that can be made waypoints, that is, local minima of the potential field. The feasible region is clearly strictly smaller than the triangle defined by the active actuators.

For a given waypoint $\overline{\mathbf{x}}$, $C(\overline{\mathbf{x}})$ is the *capture region* of a waypoint as the set of all points $\mathbf{x} \in \mathbb{R}^2$ such that, when following the direction of steepest descent from $\mathbf{x}$, one will eventually arrive at $\overline{\mathbf{x}}$. Some examples of capture regions for various triangles and waypoints are given in Figure 7.2.

For a point $\bar{\mathbf{x}}$ to be a local minimum of the potential field, the gradient $\frac{\partial U}{\partial \mathbf{x}}$ at $\bar{\mathbf{x}}$ should be zero, and the Hessian matrix $\frac{\partial^2 U}{\partial \mathbf{x}^2}$ should be positive definite. This means

that the actuator amplitudes $a_i, a_j, a_k > 0$ must be chosen such that

$$\frac{\partial U}{\partial \mathbf{x}}(\bar{\mathbf{x}}) = \begin{bmatrix} \frac{\mathbf{x_i}-\bar{\mathbf{x}}}{|\bar{\mathbf{x}}-\mathbf{x_i}|^3} & \frac{\mathbf{x_j}-\bar{\mathbf{x}}}{|\bar{\mathbf{x}}-\mathbf{x_j}|^3} & \frac{\mathbf{x_k}-\bar{\mathbf{x}}}{|\bar{\mathbf{x}}-\mathbf{x_k}|^3} \end{bmatrix} \begin{bmatrix} a_i \\ a_j \\ a_k \end{bmatrix} = \mathbf{0} \tag{7.6}$$

The space of a_i satisfying Equation (7.6) is a one-dimensional vector space, but since both Equation (7.6) and the signature of the Hessian are scale invariant, checking whether $\bar{\mathbf{x}}$ is in the feasible region reduces to 1) checking whether the elements of any solution vector a_i of Equation (7.6) have equal sign (all positive or all negative), and if so 2) checking whether for a choice of positive a_i the Hessian at $\bar{\mathbf{x}}$ is positive definite.

In the global control law described in the following section, we choose a specific point in each triangle to be the waypoint, namely the *incenter*. The *incenter* of a triangle is the center of its inscribed circle, or equivalently, the intersection of the three angle bisectors of the triangle's vertices (Figure 7.3). In extensive simulation of many different triangles shapes, the incenter was always found to be in the feasible region, and when chosen as the waypoint, the capture region of the incenter was found to generally be larger than other centers, including the centroid. While the incenter has performed well, other ways of determining waypoints can be considered. A formal proof of these favorable properties of the incenter is the subject of future work.

7.3.2 Global control using switching potential field

To extend the previously described static local control law from an actuator triplet to a full actuator network, we define a roadmap that robustly guides the robot from its start location, via the incenters of successive triangles defined by actuator triplets in the network, to its end location. The steps in the algorithm are as follows (Algorithm 5):

1. Compute all $\binom{n}{3}$ triangles that can be generated by the n actuators in the network.

2. Compute the incenters of the triangles and designate these incenters as vertices in a graph.

3. For every pair of vertices (v_1, v_2) in the graph, add a directed edge from v_1 to v_2 if v_1 is in the capture region of the potential field with local minimum at v_2. Use the robot motion model $R(\mathbf{v})$ to compute the probability $P(v_2|v_1)$ that the robot moves from v_1 to v_2 in this potential field. Set the edge weight to be the negative log of this probability: $-\log P(v_2|v_1)$.

4. The weighted graph forms a roadmap for the robot. Add the start location and end location to the graph and run Dijkstra's algorithm [111] to obtain the optimal path from start to goal, or, if no such path exists, the empty sequence.

The resulting shortest path is a sequence $\{v_i\}$ of vertices, or, with the exception of the start and end location, a sequence of incenters. We can robustly drive the robot from incenter to incenter by successively switching the amplitudes such that the next incenter v_{i+1} in the path becomes the new waypoint. Since the point v_i is in the capture region of v_{i+1}, the robot will be driven to the end location. Even though no position sensing mechanism for the robot is used, and even though the robot model contains stochastic components, the convergent nature of the potential fields will ensure that the position motion uncertainty does not grow unbounded. As long as the actuators do not move, step 4 can be repeated using the same roadmap to solve multiple queries for different start and end locations.

Algorithm 5 The Actuator Network Algorithm

1. triangles ← computeTriangles(actuatorLocations)
2. vertices ← computeIncenterLocations(triangles)
3. graph ← computeEdgeWeights(vertices)
4. path ← computePath(startVertex, endVertex, graph)

7.3.3 Computational Complexity

In step 1 with n actuators, we explore $O(n^3)$ triangles. For step 2, it takes O(1) time to compute an incenter location and associated actuator amplitudes, thus it takes $O(n^3)$ to complete this step. For step 3 with m rejection samples per edge, there is an edge for each pair of incenters, thus this step will take $O(mn^6)$. For step 4, there are $O(n^3)$ vertices and $O(n^6)$ edges. Because Dijkstra's algorithm takes $O(|E| + |V| \log |V|)$, step 4 takes $O(n^6)$. Thus, the total runtime is $O(mn^6)$.

While this result is polynomial in m and n, for certain applications requiring very fast construction of actuation strategies for very large actuator networks, it may be desirable to further reduce this runtime. The most immediate way to reduce the runtime is to not consider all possible $\binom{n}{3}$ triangles. Instead, we can modify step 1 in Section 7.3.2 only use triangles of reasonable size (not too small or too large) or discard triangles for which the capture region is fully contained in the capture region of other triangles.

It is also important to note that the computation of the roadmap is an *offline* procedure that must be carried out only once during the preparation of the roadmap for a given actuator network. In addition, many implementations of an actuator network may self-correct for an increased number of actuators by providing additional resources for computation along with each actuator. The (notably parallel) problem of computing edge weights could be solved using distributed computation across the computation elements.

7.3.4 Implementation aspects

To compute the motion probabilities for the edge weights in step 3 of the algorithm, we perform rejection sampling with m samples. For an edge from v_1 to v_2, we compute the robot motion from v_1 by integrating the robot velocity $R(\mathbf{v})$ using Euler integration. If, after a certain integration interval τ the robot is within some small distance ϵ of v_2, we consider the motion successful, and failure otherwise. Thus, each edge's weight is determined by the percentage of successful transitions from v_1 to v_2. Valid values for ϵ depend on the specific robot's size relative to the workspace and sensitivity to motion uncertainty, while τ depends strictly on the size of the region and the natural velocity of the robot. For instance, τ may be defined as the maximum amount of time required for a robot to move linearly between any two points in the planar region at its minimum velocity with no motion uncertainty, and ϵ may be set to 1% of the minimum distance between any two actuators. Tighter bounds are possible for faster movement. Even though v_1 may be in the capture region of v_2, sensor and actuator uncertainty (as captured by $R(\mathbf{v})$) can cause the robot to move temporarily outside the capture region, after which it will diverge and not succeed in reaching the current waypoint. The probability of success is determined by the fraction of the samples that successfully reach the waypoint.

The weights of the edges are taken to be the negative logarithm of the probability of success. The probability of successfully reaching the end location along a certain path is equal to the product of the probabilities of successfully moving along the edges of the path. Since multiplying probabilities P_i is equivalent to adding log-probabilities $\log P_i$, maximizing the probability of success along a path is equivalent to minimizing the sum of the negative logs of the probabilities along a path. Thus, we can efficiently compute the path with the maximum probability by using Dijkstra's algorithm from $\mathbf{p_0}$ to $\mathbf{p_e}$ using the negative log of the probability at each edge.

7.4 Simulation experiments

All simulation experiments were implemented in Matlab and executed on PCs with a 2.0GHz Intel processor and 2GB of RAM.

We fixed the workspace to be 5×5 units, and used $m = 10$ rejection samples. We explored the algorithm's effectiveness under a variety of actuator location distributions (Section 7.4.1) and motion uncertainty models (Section 7.4.2). For every choice of actuator network and robot uncertainty model, we randomly chose $k = 100$ start and end locations in the workspace, computed the optimal paths using Algorithm 5, and simulated the motion of the robot in the actuator network.

For a given actuator network topology, we compute the average probability of the robot successfully traveling from a random start location to a random end location.

7.4.1 Varying Actuator Placement

We randomly placed $n \in \{5, \ldots, 10\}$ actuators throughout the workspace according to two distribution models. In the *bordered* distribution strategy, each actuator was placed at a location chosen uniformly at random on the border of the workspace. In the *interior* distribution strategy, actuators were scattered uniformly at random throughout the entire workspace. For each actuator placement model and for each possible number of actuators from 5 to 10, 100 random actuator geometries were produced. The robot motion model was set to have zero motion uncertainty and a probabilistic roadmap was constructed according to the above algorithm.

The results are shown in Figure 7.4. We examine the two cases where 1) the start and end locations are within the convex hull of the actuators and 2) the start and end locations are anywhere in the workspace. In the first case, the data suggests that the border selection strategy performs just as well as the interior selection strategy.

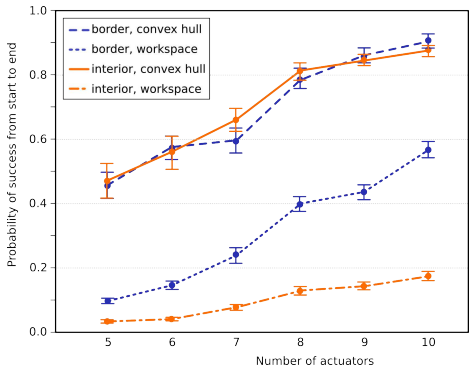
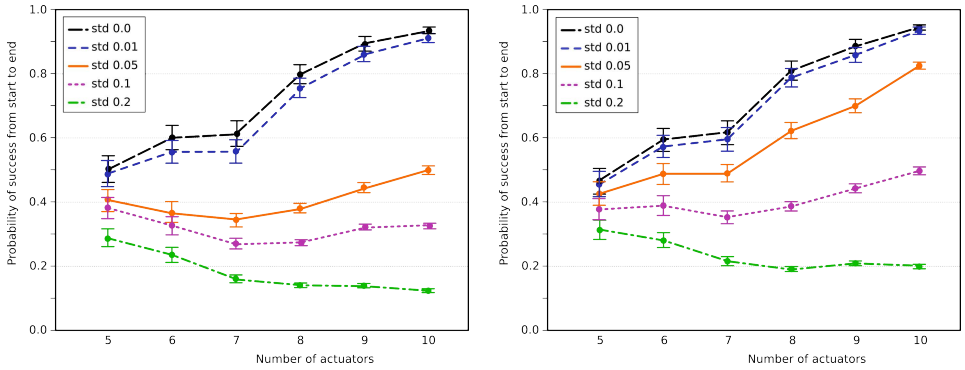


Figure 7.4. Simulation result averaging over 100 trials of an actuator network and 100 start-end location pairs, with $n = 5 \ldots 10$ actuators and two placement strategies: border placement and interior placement. Results for each are shown with start and end locations chosen randomly across the entire workspace or only within the convex hull of the actuators. These simulations used 0.01 standard deviation polar motion uncertainty.

Because the convex-hull eliminates start/end locations that are outside the convex hull and therefore impossible to reach, we can see that there is no robustness advantage for one method over another. As the algorithm performs comparably in the convex-hull restricted test, the border-selection method is better in the full-workspace experiment, because on average, its convex hull will cover a larger area, which in turn means more start/end locations will be reachable.

As would be expected, increasing the number of actuators increases the probability of success with the workspace model. We can see that the probability of success also improves as we increase actuators for the convex-hull method. We discuss this property further in Section 7.4.2.

The simulation results suggest lower bounds on the effectiveness of smart actuator placement strategies. Better results can be obtained by (deterministically) optimizing the placement of actuators to 1) maximize the area of the workspace that falls into the capture region of at least one triangle, and 2) maximize the connectivity between points in the workspace, particularly when likely start and end locations are known in advance. Such an optimal-placement algorithm will be the subject of future research.



(a) Average success rate under Cartesian motion uncertainty.

(b) Average success rate under polar motion uncertainty.

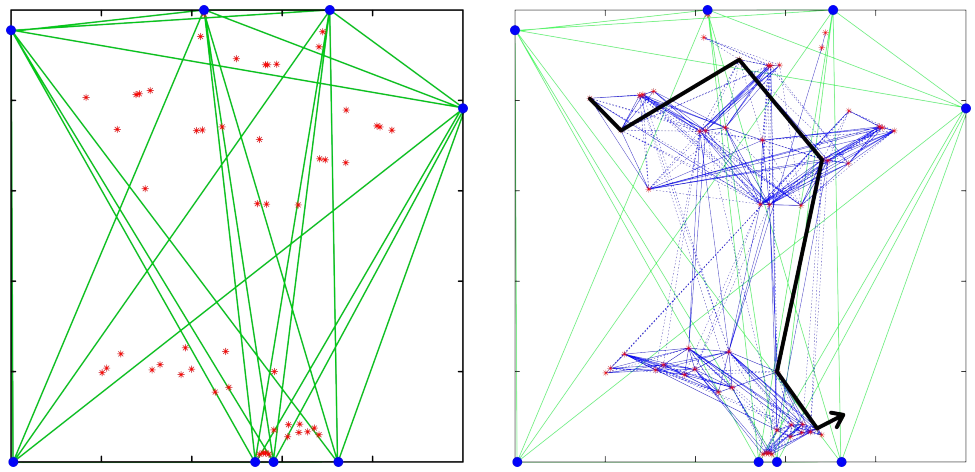
Figure 7.5. Comparison of the results of varying Gaussian motion uncertainty for a fixed set of actuator locations under two motion uncertainty models, with start and end locations chosen within the convex hull of the actuators.

7.4.2 Varying Motion Uncertainty

For networks of $n \in \{5, \ldots, 10\}$ actuators, we examined how the probability of successful completion varied for both the Cartesian motion uncertainty model described in Equation (7.4) and the polar motion uncertainty model described in Equation (7.5). We experiment with different errors (standard deviations $\sigma \in \{0, 0.01, 0.05, 0.1, 0.2\}$). The results are summarized in Figure 7.5, and Figure 7.6 shows an example of the roadmap generated by the algorithm.

The figure shows that increasing the motion uncertainty will result in reduced probability of success, for both motion uncertainty models and any number of actuators. Under large motion uncertainty, the robot is more likely to drift outside the capture region, resulting in failure to reach the next waypoint and hence to successfully complete the path to the end location.

Because these examples are only of start/end locations within the convex hull, adding actuators has two effects. As we increase the number of actuators, the area of the convex hull will become larger, which means that the average path length be-



(a) Triangles and corresponding incenters (denoted $*$) generated from 8 randomly placed actuators.

(b) Roadmap showing edges between incenters and an example path (thick line) through the network.

Figure 7.6. Example of a simulation of the actuator-networks algorithm with $n = 8$ actuators and Cartesian motion uncertainty with $\sigma = 0.01$. The actuators are placed randomly on the border of a square workspace, the incenters of all possible triangles between them form vertices in a roadmap with edges containing the probability of successful transition by activation of an actuator triplet.

tween start/end goals in the convex hull increases, making the effects of the motion uncertainty more significant. More actuators also means more flexibility in the number of paths, due to an increased number of incenters and overlapping triangles. As we increase the number of actuators, the incremental addition to the convex hull will decrease, and the number of additional waypoints will grow quadratically with the number of new actuators. Thus, for larger motion uncertainty models, the probability of success first decreases and then increases. This effect is more extreme depending on how significant the motion uncertainty is.

For 10 actuators, standard deviations of 0.0, 0.01, 0.05, 0.1, and 0.2, and Cartesian motion uncertainty, the average probabilities of success are 94.5%, 93.4%, 82.5%, 49.6%, and 19.8%. For Polar motion uncertainty, the average probabilities of success are 93.3%, 91.0%, 49.8%, 32.4%, and 12.2%.

7.5 Conclusions and Future Work

We consider the problem of localization-free guidance of a robot using an actuator network of beacons for use in steering simple, low-cost robots. The Actuator Networks system and algorithm steer unmonitored robots between points using an external network of actuators and a probabilistic roadmap. Our algorithm was able to produce relatively high probabilities of successful navigation between randomly-selected points even in the presence of motion uncertainty.

The low number of actuators necessary for successful steering in our technique has important consequences for the robustness of these methods in practice. An inexpensive way to guarantee continuous operation of an actuator network is to use more than the minimum number of actuators required for high-probability performance; as an example, with 20 actuators under border placement and 1% Cartesian motion uncertainty, even if half of the actuators eventually fail, the probability of completion would not drop significantly.

We plan to explore several extensions in future work. The technique of actuator networks can be extended to consider obstacles in the workspace. An obstacle can affect an Actuator Network in three ways: (1) the obstacle restricts the motion of the mobile robot in the workspace, (2) the obstacle blocks the signal from an actuator, and (3) the obstacle causes multi-path effects as the signals from the actuators reflect off the obstacles. To model case 1, we can represent obstacles implicitly in the graph via the edge weights encoding transition success probabilities. As discussed in Section 7.3.4, we estimate the probability $P(v_2|v_1)$ of successfully moving from vertex v_1 to vertex v_2 by simulating the robot's motion as it follows the gradient of the signal generated by the actuators. If the mobile robot's motion intersects an obstacle during a simulation, we determine that a failure has occurred in our rejection sampling step. For case 2, we can modify the simulation so the gradient used by the mobile robot

does not include signal from a particular actuator if a line segment between that actuator and the mobile robot's current location intersects an obstacle. Since the probability of success for edges in the graph will decrease when obstacles are present, the number of actuators necessary in order to find a feasible plan will increase. Case 3 is known to be difficult to model effectively, and is a significant problem for certain domains such as RSSI localization. As future work, we can also evaluate how different multi-path models will affect the robot by including this in the determination of the edge-weights in a similar fashion as case 2.

We would also like to explore the alternative problem of designing an algorithm for placement of actuators.

Chapter 8

Privacy: Respectful Cameras

8.1 Introduction

Since September 11, 2001, security concerns have led to increasing adoption of surveillance systems, raising concerns about "visual privacy" in public places. New technologies allow for the capture of significantly more detailed information than the human eye can perceive. Surveillance technologies are additionally empowered by digital recording, allowing footage to be stored indefinitely, or processed and combined with additional data sources to identify and track individuals across time and physical spaces. Robotic cameras can be servoed to capture high resolution images over a wide field of view. For example, the Panasonic KX-HCM280 pan-tilt-zoom camera costs under $750 and has a built-in web-server and a 21x optical zoom (500 Mpixels per steradian). The applications of these surveillance technologies extends beyond security, to industrial applications such as traffic monitoring and research applications such as observing public behavior.

McCahill et al. estimate that there are approximately 4 million public surveillance cameras deployed in the UK [126]. The U.S. has also deployed a number of camera

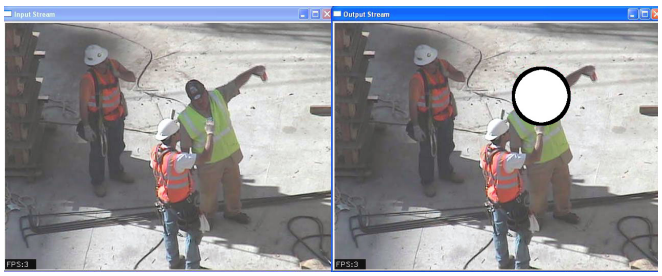


Figure 8.1. A sample video frame is on left. The system has been trained to track green vests such as the one worn by the man with the outstretched arm. The system output is shown in the frame on the right, where an elliptical overlay hides the face of this man. The remainder of the scene including faces of the other two workers wearing orange vests, remain visible. Note how the system successfully covers the face even when the vest is subjected to a shadow and a partial occlusion. Please visit "http://goldberg.berkeley.edu/RespectfulCameras" for more examples including video sequences.

systems in cities such as New York, Chicago, and San Francisco for public monitoring [11, 139, 135, 101]. Deployments of such large-scale government-run security systems, in conjunction with numerous smaller-scale private applications, raise fundamental privacy concerns.

In this chapter, we explore an approach we call "Respectful Cameras", that allows monitoring of activity but hides the faces of people who choose to wear recognizable markers such as hats or vests that are made available. We show an example of the Respectful Cameras system's output in Fig 8.1. The system allows human actions to be observable so that people can monitor what is going on (i.e., at a construction site or airport terminal) for security or public relations purposes. We envision such a system being made widely available, as these markers would be cheap, unobtrusive, and easily mass-produced. For example, we could provide inexpensive hats of a particular color or pattern at the border of the space where cameras are present, similar to the respectful hats or leg-coverings that are made available at the entrance of churches and synagogues.

Existing face and people tracking methods have difficulty tracking in real-time un-

der moving backgrounds, changing lighting conditions, partial occlusions and across facial orientations. We investigate a new approach that uses markers worn by individuals to improve the robustness of face or person detection required for obscuring individual identity, providing a method for individuals to opt-out of observation. These markers provide a visual cue for our system by having the features of the marker such as color, size, and shape to be distinguishable from the background. We use the location of the marker to infer the location of the faces of individuals who wish to "opt-out" of observation.

Recent advances in computer processing have made our algorithms utilizing Adaptive Boosting (AdaBoost) and Particle Filtering feasible for real-time applications. Our approach learns a visual marker's color-model over a 9-dimensional RGB HSV LAB color space and uses Adaptive Boosting to detect a marker in a single image. We then extend this approach using a probabilistic reformulation of AdaBoost so that we can apply Particle Filtering to integrate temporal information. Results suggest that our implementation can track markers and keep false negative rates below 2%.

8.2 Providing Visual Privacy

Protecting the privacy of individuals has become increasingly important as cameras become more ubiquitous and have greater capabilities, particularly resolution and zoom. Examples of this interest includes The National Science Foundation (NSF) funding of TRUST [3], a research center for security and privacy, and privacy has been the subject of recent symposia such as Unblinking [5].

Changes in surveillance ubiquity and capabilities raise questions about the fair balance of police power (the inherent authority of a government to impose restrictions on private rights for the sake of public welfare, order, and security) to monitor public

places versus the citizens' freedom to pass through public spaces without fear of government monitoring. According to Gavison, a loss of privacy occurs through visual surveillance by the extent we are known by others and subject to their attention [72]. He discusses our expectation that our actions are observable only by those we see around us, and thus we can judge how we should act. Nissenbaum describes how the high-resolution and zooming capabilities of cameras applied to visual surveillance also violate the contextual expectations of how people will be perceived in public [141]. This places the burden upon an individual to conduct himself or herself as if every move could be recorded and archived. Finally, it should be noted that it is not just surveillance that threatens privacy, but also the ability to be identified [172].

Researchers such as Hampapur et al. have analyzed the system architecture requirements of surveillance systems and built such a system [79]. Chen et al. built a system for people detection, tracking and recognition [32] using gait analysis, face detection and face recognition. As visual privacy is a significant concern of such systems, a number of researchers have explored how to integrate privacy systems into these surveillance systems [170, 220, 33, 29, 58]. Others such as Chinomi et al. have evaluated different methods for obscuring people in video data [35]. Google has started experimenting with automated face blurring in Google Street View [75].

8.3 Our Motivation: Demonstrate Project

Prior to the work in this Chapter, in the Fall of 2004, to commemorate the 40th anniversary of the Free Speech Movement, many researchers in Berkeley's Automation Lab, placed a robotic camera on top of the Martin Luther King Building overlooking Sproul Plaza, the center of Berkeley's campus. The camera was made publicly accessible over the internet at demonstrate.berkeley.edu. Visitors to the website could instruct the camera to pan, tilt and zoom in on any region of the plaza and then, if

desired, view a live video feed, take photos of particularly interesting scenes and comment on them. These photos and initial comments often sparked further discussion through follow up comments. The goals of the project was two fold. Firstly, it was to produce an art piece that spread public awareness about the capabilities of robotic cameras and about what surveillance technologies are available and in use in public spaces today. Secondly, it was to publicize and stress-test the dissertation project of Dr. Song [180], which was concerned with new ways for simultaneous collaborative control of a shared robotic camera by many users.

When the project started, we had no idea of the privacy concerns and debate that the project would unleash. The day before the project was supposed to launch, the camera mysteriously was unavailable. After some quick analysis, we deduced that the wire transmitting the video was sabotaged, snipped and carefully concealed. The problem was promptly corrected, and launched as scheduled. The project quickly became an object of debate and concern among university administrators, even though the camera surveilled a very public space that formed the center of Berkeley's campus. We were happy with the results of the project, as we had achieved our goal of starting conversations between diverse groups such as students, administrators and lawyers, who all started to analyze the ramifications of surveillance and lack of visual privacy. These discussion lead to the technical problem of providing automated privacy in surveillance systems, which we describe in this chapter.

8.4 System Input

Our system relies on visual markers worn by individuals who wish to have their face obscured. Our input is the sequence of images from a video stream. Let i be the frame number in this sequence. Each image consists of a pixel array where each pixel has a red, green, and blue (RGB) component.

8.5 Assumptions

We use the location and form-factor of each visual marker as a proxy for a location and form-factor of a corresponding human head. Thus, we assume there is an offset between the marker's location and the estimated face's location. Similarly, we assume the face's size will be proportional to the size of the visual marker. Intuitively, this means that as the marker's size shrinks, the face will shrink proportionally.

We make the following additional assumptions:

- Whenever a person's face is visible, then the visual marker worn by that person is visible.
- In each frame, all visible markers have a minimum number of visible, adjacent pixels

8.6 System Output

Our objective is to place solid ellipses to obscure the face of each individual wearing a marker, while minimizing the overlay area to allow observation of actions in the scene.

For each frame in the input stream, the system outputs a set of axis-aligned elliptical regions. These regions should completely cover all faces of people in the input image who are wearing markers. The ith output image has a set of elliptical regions E_i associated with it. Each element in E_i is defined by a center-point, denoted by an x and y position, and major and minor axis r_x and r_y:

$$E_i = \{(x, y, r_x, r_y)\} \tag{8.1}$$

The ith output video frame is the same as the ith input frame with the corresponding regions E_i obscured via solid ellipses.

Failing to detect a marker when one is present (false negative) is worse than placing an ellipse where there is no face (false positive).

8.7 Three Phases of System

Our solution consists of three phases: (A) offline learning of a statistical classifier for markers, (B) online marker detection and (C) online marker tracking.

8.7.1 Phase A: Offline Training of the Marker Classifier

We train a classifier offline, which we then use in the two online phases. For classification, we use the statistical classifier, AdaBoost, which performs supervised learning on labeled data.

Input and Output

A human "supervisor" provides the AdaBoost algorithm with two sets of samples as input, one for pixels colors corresponding to the marker T_+ and one for pixels colors corresponding to the background T_-. Each element of the set has a red value r, a green value g, a blue value b and the number of samples with that color m. Thus, the set of colors corresponding to marker pixels is

$$T_+ = \{(r, g, b, m)\} \tag{8.2}$$

and the sample set of pixels that correspond to background colors

$$T_- = \{(r, g, b, m)\} \tag{8.3}$$

As we are using a color-based method, the representative frames must expose the system across all possible illuminations. This includes maximum illumination, minimal illumination, the object under a shadow, and any potential hue effects caused by lighting phenomena such as a sunset. We discuss the AdaBoost formulation in more detail in Section 8.8.1.

We use a Probabilistic AdaBoost formulation that produces a strong classifier $\eta : \{0, \ldots, 255\}^3 \mapsto [0, 1]$. This classifier provides our output, a prediction of the probability that the RGB color of any pixel corresponds to the marker color.

8.7.2 Phase B: Online Static Marker Detector

For static detection, each frame is processed independently. This phase can be used on it's own to determine marker locations, or can be used as input to a dynamic method such as what we describe in Phase C.

Input and Ouput

The Marker Detector uses as input the model generated from AdaBoost, as well as a single frame from the video stream.

We can use the marker detector without tracking to infer the locations of faces. This would produce for the ith image, a set of regions E_i as defined in Section 8.6, to obscure each face. We represent the state of each marker in the ith image as a bounded rectangle. We denote the set of rectangular bounding regions for each marker in image i as R_i. Each rectangular region is represented by a center-point, denoted by an x and y position, and it's size, denoted by a width Δx and a height Δy:

$$R_i = \{(x, y, \Delta x, \Delta y)\} \tag{8.4}$$

There is an explicit mapping between the size and location of the bounding regions R_i and the sizes and locations of elliptical overlays E_i. The rectangles in R_i are restricted by the assumptions described in Section 8.5, but have the flexibility to change its shape as the marker moves around the observed space. When used as a component of a marker tracker, the detector supplies the same set of rectangles for initializing the tracker, but also determines for each pixel the probability $P(I_i(u, v))$ that each pixel (u, v) corresponds to the visual marker in image I_i.

8.7.3 Phase C: Online Dynamic Marker Tracker

The dynamic marker tracker uses temporal information to improve the Online Detector. We do this by using information from the Static Detector along with a Particle Filter for our temporal model.

Input and Output

The dynamic marker tracker uses both the classifier determined in the training phase and output from the static image recognition phase. We process a single frame per iteration of our Particle Filter. Let the time between the previous frame and the ith frame be $t_i \in \mathbb{R}_+$, and the ith image be I_i. We discuss Particle Filtering in more depth in Section 2.4.3, but it requires three models as input: a prior distribution, a transition model, and an observation model. We use the Static Marker Detector to initialize a Particle Filter for each newly-detected marker. We also use the probabilities $P(I_i(u, v))$, supplied by the Static Marker Detector, to determine the posterior distribution of a marker location for each Particle Filter, given all previously seen images.

The output for the ith frame is also the set of regions E_i as defined in Section 8.6.

8.8 Phase A: Offline Training of the Marker Classifier

To train the system, a human "supervisor" left-clicks on pixels in a sample video to add them to the set T_+, and similarly right-clicks to add pixels to set T_-.

In this phase, we use the two sets T_+ and T_- to generate a strong classifier η, which assigns the probability that any pixel's color corresponds to the marker, providing $P(I_i(u, v))$. Learning algorithms are designed to generalize from limited amounts of data. For instance, with the AdaBoost algorithm, we needed a thousand labeled training samples. Also, as our classification algorithm is linear in the number of dimensions of our dataset (9 in our formulation) and number of hyperplanes used as weak hypotheses in the model (20 in our experiments), we can evaluate this classifier in realtime.

8.8.1 Review of AdaBoost

AdaBoost uses a set of labeled data to learn a classifier. This classifier will predict a label for any new data. AdaBoost constructs a strong classifier from a set of weak hypotheses.

Let X be a feature space, $Y \in \{-1, 1\}$ be an observation space and $\{h : X \to Y\}$ be a set of weak hypotheses. AdaBoost's objective is to determine a strong classifier $H : X \mapsto Y$ by learning a linear function of weak hypotheses that predicts Y given X. At each iteration from $t = (1 \dots T)$, AdaBoost incrementally adds a new weak hypothesis h_t to the strong classifier H:

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{8.5}$$

and

$$H(x) = \text{sign}\left(f(x)\right) \tag{8.6}$$

Let $\eta(x) = P(Y = 1|X = x)$, and define AdaBoost's loss function $\phi(x) = e^{-x}$. The objective of AdaBoost is to minimize the expected loss or

$$E(\phi(yH(x))) = \inf_{H} \left[\eta(x)\phi(H(x)) + (1 - \eta(x))\phi(-H(x))\right] \tag{8.7}$$

This is an approximation to the optimal Bayes Risk, minimizing $E[l(H(X), Y)]$ with loss function

$$l(\widehat{Y}, Y) = \begin{cases} 1 & \text{if } \widehat{Y} \neq Y \\ 0 & \text{otherwise} \end{cases} \tag{8.8}$$

To determine this function, we use a set of training data $\{(x_i, y_i)|x_i \in X, y_i \in Y\}$ sampled from the underlying distribution.

AdaBoost is an iterative algorithm where at each step, it integrates a new weak hypothesis into the current strong classifier, and can use any weak hypothesis with error less than 50%. However, we use the greedy heuristic where at each iteration, we select a weak hypothesis that minimizes the number of incorrectly labeled data points [199]. We use a standard AdaBoost classifier of this form in Phase B, trained on our labeled data.

Recasting Adaboost to Estimate Probabilities

Typically, as described in [164], AdaBoost predicts the most likely label that an input will have. Friedman et. al describe how to modify the AdaBoost algorithm to produce a probability distribution, rather than a binary classification[68]. The modified strong classifier determines the probability that an input corresponds to a label of 1 (as opposed to -1) and is defined by

$$\eta(x) = P(Y = 1|X = x) = \frac{e^{2f(x)}}{1 + e^{2f(x)}} \tag{8.9}$$

We use this probabilistic formulation to determine the probability that each pixel corresponds to a marker, as $\eta(x) = P(I_i(u, v))$, in Phase C.

8.8.2 Determining Marker Pixels

We begin by applying Gaussian blur with standard deviation σ_I to the image, which enhances robustness to noise by integrating information from nearby pixels. We use these blurred pixels for T_+ and T_-. We then project our 3 dimensional RGB color space into the two additional color spaces, Hue, Saturation, Value (HSV) [63] and LAB [91] color-spaces. HSV performs well over varying lighting conditions because Value changes over varied lighting intensities, while Hue and Saturation do not. LAB is designed to model how humans see color, being perceptually linear, and is particularly well suited for determining specularities. This projection of RGB from T_+ and T_- into the nine-dimensional RGBHSVLAB color space is the input to AdaBoost.

For weak hypotheses, we use axis-aligned hyperplanes (also referred to as decision stumps),each of which divides the space along one of the nine dimensions into two sets. These hyperplanes also have a +/- direction, where all 9-dimensional tuples that are in the direction and above the hyperplane are labeled as visual marker pixels, and all other tuples are non-marker pixels. The hyperplane separating dimension d at a threshold j is described by:

$$h_{d,j}(X) = \begin{cases} 1 & \text{if } X[d] \geq j \\ -1 & \text{otherwise} \end{cases} \tag{8.10}$$

Our set of weak hypotheses also includes the complement of these hyperplanes $\overline{h_{d,j}}(X) = -h_{d,j}(X)$. By projecting the initial RGB space into the additional HSV and LAB spaces, we provide more classification flexibility as we have more weak classifiers. For the weak hypothesis, AdaBoost chooses the dimension and threshold

at each round that minimizes the remaining error. The algorithm terminates after running for some constant number, n, iterations. There are many ways to set n, for instance splitting the data into a learning set and a validation set. In this technique, learning is applied to the learning set, and the generalization accuracy is evaluated on the validation set. Thus, we can observe how our model performs not on the data it is exposed to, but at predicting other unobserved data.

Rather than using the learning set and validation set directly, we went through a calibration process where we recorded video, exposing the marker to all areas in the viewing range of the camera and over varied lighting conditions and marker orientations. We went through the process of collecting an initial set of 100 marker pixels and 100 background pixels, and varied n from 5 to 50. If there were still patches of the background or marker that were misclassified, we would add these mistakes to the model, and repeat the learning. For instance, if a portion of the background was misclassified as the marker, we would add some of those pixels to the background set, and repeat the learning. This iterative approach provided an intuitive way for the user to distinguish between the marker and the background.

8.9 Phase B: Online Static Marker Detector

This section describes our marker detection algorithm, using only the current frame. Once we have the strong classifier from AdaBoost, we apply the following steps: (1) Apply the same Gaussian blur to the RGB image as we did for training. (2) Classify each pixel in the image (3) Cluster marker pixels (4) Select all clusters that satisfy the constraints.

8.9.1 Clustering of pixels

To determine the pixels to marker correspondence, we apply the connected-component technique [160]. The connected component algorithm is applied to an image with two types of pixels, foreground and background pixels, or in our case, marker and non-marker. Connected component will recursively group adjacent foreground pixels into the same cluster. We assign a unique group id to each cluster's pixels. Thus, a connected group of marker pixels would be determined to correspond to the same marker as they have the same group id. This yields a set of marker pixels for each visual marker in the frame.

To remove false positives, we enforce additional constraints about each identified marker cluster. We verify there are at least ζ_1 pixels in the cluster, and that ratio of width (Δx) to height (Δy) falls within a specified range from ζ_2 to ζ_3:

$$\zeta_2 \leq \frac{\Delta x}{\Delta y} \leq \zeta_3 \tag{8.11}$$

Requiring at least ζ_1 pixels per cluster prunes out false positives from small areas of color in the background that do not correspond to the marker. We also found that in some backgrounds, there would be long vertical or horizontal strips similar to our marker color, which would be incorrectly labeled as a marker as it was the same color. However, our markers have a bounded ratio between width and height. Using this knowledge helps remove these false positives.

8.10 Phase C: Online Dynamic Marker Tracker

We use Particle Filtering to incorporate temporal information into our models, improving robustness to partial occlusions. Particle Filtering requires a probability distribution for the likelihood of the state given the current, indirect observations.

We provide this Observation Model by extending the theory from our Static Marker Detector from Phase B, using the probability that each pixel corresponds to a marker that is provided by the Probabilistic Adaboost formulation described in Section 8.8.1.

8.10.1 Marker Tracking

In this section, we define our transition models and our observation models for our Particle Filter. We also describe how to track multiple markers simultaneously.

Marker Model

The state of a marker is defined with respect to the image plane and is represented by a 6 tuple of a bounding box's center x and y positions, the height and width of the bounding box, orientation, and speed. As can be seen in Figure 8.2 this yields:

$$z = (x, y, \Delta x, \Delta y, \theta, s) \tag{8.12}$$

We model the marker in image coordinates, rather than world coordinates to improve the speed of our algorithms.

Transition Model

The transition model describes the likelihood of the marker being in a new state, given its state at the previous iteration, or $P(Z_i|Z_{i-1} = z_{i-1})$. Our model adds Gaussian noise to the speed, orientation, bounding-box width, and bounding box height and determines the new x and y position via Euler integration. Let $W \sim N(0, 1)$ be a sample from a Gaussian with mean zero and standard deviation of one. The mean μ and standard deviation σ for each portion of our model are set a priori.

Formally:

$$\begin{aligned} x_i &= x_{i-1} + s_i \cdot \cos(\theta_i) \cdot t_i \\ y_i &= y_{i-1} + s_i \cdot \sin(\theta_i) \cdot t_i \\ \Delta x_i &= \Delta x_{i-1} + \sqrt{t_i} \cdot (\sigma_{\Delta x} \cdot W + \mu_{\Delta x}) \\ \Delta y_i &= \Delta y_{i-1} + \sqrt{t_i} \cdot (\sigma_{\Delta y} \cdot W + \mu_{\Delta y}) \\ s_i &= \max(0, \min(s_{\max}, s_{i-1} + \sqrt{t_i} \cdot \sigma_s \cdot W)) \\ \theta_i &= \theta_{i-1} + \sigma_\theta \cdot \sqrt{t_i} \cdot W \end{aligned} \tag{8.13}$$

At each iteration, we enforce the width and height constraints for each particle described in Section 8.9.1. The sample from the Gaussian, after being scaled by μ and σ, must be rescaled according to $\sqrt{t_i}$ (as defined in Section 8.7.3) to compensate for changing frame rates.

We modify the variance of the noise added to the distributions by $\sqrt{t_i}$ is because of the following proposition.

Proposition 4. *Define r_1 and r_2 as the rates of waiting times before adding in new samples from $N(0,1)$ Gaussians into the current state, and $c \in \mathbb{N}^+$, with $r_1 = c \cdot r_2$. We must sample $\sqrt{\frac{r_2}{r_1}}N(0,1)$ at rate r_2 to preserve the same variance over time $N(0,1)$ at rate r_1.*

Proof. Define i.i.d. random variables $W_0, \ldots, W_c$ with distribution $N(0,1)$. Over a single waiting time of r_1, we have c additions of Gaussians of rate r_2. Because the Ws are linearly independent:

$$\begin{aligned} \mathsf{Var}[W_0] &= \mathsf{Var}[h \textstyle\sum_{i=1}^{c} W_i] \\ 1 &= h^2 \textstyle\sum_{i=1}^{c} \mathsf{Var}[W_i] \\ 1/\sqrt{c} &= h \end{aligned}$$

Therefore, sampling from distribution $N(0,1)$ at the rate r_1 has the same variance as sampling from $\sqrt{\frac{r_2}{r_1}}N(0,1)$ at rate r_2. ∎

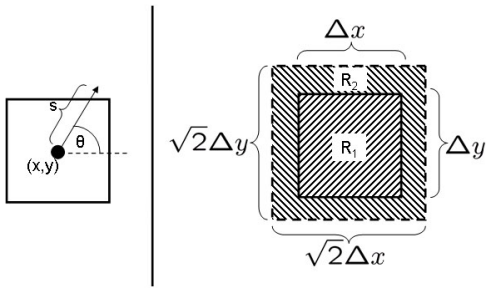


Figure 8.2. Illustrates the state of a single bounding box (left) and the probability mask used for the Particle Filter's observation model (right).

Observation Model

The observation model describes the distribution of the marker's state given an image, but our formulation gives a probability per pixel, rather than per marker state. We use an objective function as a proxy for the observation model, which has a probability of 1 if the bounding box tightly bounds a rectangular region of pixels with high probability. Let bounding box R_1 be the marker's state and bounding box R_2 have the same midpoint as R_1 but have size $\sqrt{2}\Delta x \times \sqrt{2}\Delta y$. R_1 and R_2 are disjoint. The $\sqrt{2}$ scaling factor makes the areas of R_1 and R_2 be equal. Then:

$$R_1 = \left\{ (u,v) \;\middle|\; |x-u| \leq \tfrac{\Delta x}{2}, |y-v| \leq \tfrac{\Delta y}{2} \right\} \tag{8.14}$$

$$R_2 = \left\{ (u,v) \;\middle|\; |x-u| \leq \tfrac{\Delta x}{\sqrt{2}}, |y-v| \leq \tfrac{\Delta y}{\sqrt{2}}, (u,v) \notin R_1 \right\} \tag{8.15}$$

$$P_1(Z_i = z_i | I_i) = \frac{1}{\Delta x \Delta y} \left(\sum_{(u,v) \in R_1} P(I_i(u,v)) \right) \tag{8.16}$$

$$P_2(Z_i = z_i | I_i) = \frac{1}{2\Delta x \Delta y} \begin{pmatrix} \sum_{(u,v) \in R_1} P(I_i(u,v)) + \\ \sum_{(u,v) \in R_2} 1 - P(I_i(u,v)) \end{pmatrix} \tag{8.17}$$

Our final metric used as our observation model is:

$$P(Z | E_t = e_t) = (1 - P_1)P_1 + P_1 P_2 \tag{8.18}$$

This metric has the essential property that there is an optimal size for the bounding box, as opposed to many other metrics which quickly degenerate into determining the marker region to consist of all the pixels in the image or just a single pixel. For intuition, assume the projection of the visual marker produces a rectangular region. If a particle's bounding region is too large, its objective function will be lowered in region R_1, while if it is too small, then the objective function would be lowered in region R_2. This function yields a probability of 1 for a tight bounding box around a rectangular projection of the marker, yields the probability of 0 for a bounding box with no pixels inside that correspond to the marker, and gracefully interpolates in between (according to the confidence in R_1). We illustrate the two areas in Figure 8.2.

Multiple-Object Filtering

Our formulation uses one Particle Filter per tracked marker. To use multiple filters, we must address the problems of: (1) markers appearing, (2) markers disappearing and (3) multiple filters tracking the same marker. We make no assumptions about where markers can be obstructed in the scene.

For markers appearing, we use the output of the Marker Detection algorithm to determine potential regions of new markers. We use an intersection over minimum (IOM) metric, defined for two regions R_1 and R_2 is:

$$IOM(R_1, R_2) = \frac{\text{Area}(R_1 \cap R_2)}{\min(\text{Area}(R_1), \text{Area}(R_2))} \tag{8.19}$$

If a Marker Detection algorithm has an IOM of more than a specified overlap γ_1 with any of Particle Filter's most likely location, then a Particle Filter is already tracking this marker. If no such filter exists, we create a new marker at this region's location by creating a new Particle Filter with the location and size of the detection region. We choose an orientation uniformly at random from 0 to 2π, and choose speed from 0 to the maximum speed $s_{\max}$ that is chosen a priori.

To address disappearing markers, we require that the probability of the state of at least one particle for a filter exceeds γ_2, otherwise the filter is no longer confident about the marker's location, and is deleted.

Multiple Particle Filters can become entangled and both track the same marker. If the IOM between two Particle Filters' exceeds the same threshold as appearing filters γ_3, we remove the filter that was created most recently. We remove the most recent to maximize the duration a Particle Filter tracks its marker.

8.11 Experiments

We ran two sets of experiments to evaluate performance. We experimented in our lab where we could control lighting conditions and we could explicitly setup pathological examples. We then monitor performance on video from a construction site as we vary model parameters. All tests involved video from a Panasonic KX-HCM280 robotic camera, transmitting an mJPEG stream of 640x480 images. We ran all experiments on a Pentium(R) CPU 3.4 GHZ.

Currently, the system has not been optimized, and we could easily extend our formulation to incorporate parallelism. The rate that we can process frames is about 3 frames per second, which is approximately 3x slower than the maximum incoming frame rate of 10fps.

For both the Lab Scenario and Construction Site, we trained the AdaBoost algorithm on 2 one-minute video sequences specific to the environment, using the method described in Section 8.7.1, exposing the system to many potential backgrounds, location and orientations of the visual markers, and over all lighting conditions that the experimental data experiences. After training AdaBoost, we used the same sequences

to calibrate our parameters. In our experiments we used:

$\sigma_{\Delta x}$	$= 25$ pixels	$\sigma_{\Delta y}$	$= 25$ pixels
σ_s	$= 100$ pixels	σ_θ	$= \frac{3}{2}\pi$ radians
$\mu_{\Delta x}$	$= 12.5$ pixels	$\mu_{\Delta y}$	$= 12.5$ pixels
$s_{\max}$	$= 300$ pixels	σ_I	$= 3.7 pixels$
ζ_1	$= 300$ pixels	ζ_2	$= \frac{1}{5}$
ζ_3	$= 5$	γ_1	$= 0.2$
γ_2	$= 0.4$	γ_3	$= 0.2$
# Particles	$= 2000$	# Weak Hypotheses	$= 20$

We define an image to be a false negative if any part of any face is visible and to be a false positive if there is an obscuring region in E_i that does not touch any of the faces in image i. These metrics are independent of the number of people in the scene.

To determine the statistics for each experiment, we processed each video with our system and then went through each image, frame by frame, and hand labeled each for false positives and false negatives. We then went through the same sequence twice more to help ensure quality results. This required approximately 30 seconds per frame, or nearly 60 hours of manual labeling for the experiments presented.

8.11.1 Lab Scenario Experiments

Within the lab, where we can control for lighting changes, we explore scenarios that challenge our system. Our marker is a yellow construction hat, and we assume the face is at the bottom-center of the bounding box and the same size as the hat. We evaluate how the system performs when 1) there are lighting conditions that the system never was trained on, and 2) two individuals (and their respective markers)

Figure 8.3. Example of a False Negative with the Respectful Cameras System: A sample image frame input on left image, with output regions overlayed on right image. This sample illustrates where an intense light from a flashlight induced a specularity, causing the classifier to lose track of the hat. As a result, the right image has no solid white ellipses overlaid on the face as it should.

cross. Lab experiments were run on 51 seconds of data acquired at 10 frames per second (fps). We summarize our results in the following table:

Lab Scenario Experiments

Experiment	# Frames	Correct	FPs	FNs	FP+FNs
Lighting	255	96.5%	0.0%	3.5%	0.0%
Crossing	453	96.9%	0.0%	3.1%	0.0%

Table 8.1. This tables shows the performance of in-lab experiments. To evaluate the system, we place each frame into the category of correctly obscuring all faces without extraneous ellipses, being a false negative but not false positive, being a false negative but no a false positive, and being both a false negative and false positive. We denote false negatives with FN and false positives with FP.

Lighting

In this setup, there is a single person, who walks past a flashlight aimed at the hat during two different lighting conditions. We experiment with all lights being on, and half of the lab lights on. In the brighter situation, the flashlight does not cause the system to lose track of the hat. However, in the less bright situation, the hat gets washed out with a specularity and we fail to detect the hat during this lighting problem. An explanation for why the specularity only was visible in the less bright situation is that our camera dynamically modifies the brightness of the image

Figure 8.4. Example of Crossing Markers and the Respectful Cameras System: A sample image frame input on left image, with output regions overlayed on right image. This sample illustrates tracking during a crossing, showing how the Particle Filter grows to accommodate both hats.

Figure 8.5. Example of the Respectful Cameras System: A sample image frame input on left image, with output regions overlayed on right image. This sample illustrates tracking after a crossing (one frame after Figure 8.4), showing how the system successfully creates a second filter to best model the current scene.

depending on the scene it observes. Thus, in the darker scene, the specularity would have been perceived to be brighter than in the brighter scene. We show one of the failing frames in Figure 8.3. In general, the system performs well at interpolating between observed lighting conditions, but fails if the lighting is dramatically brighter or darker than the range of lighting conditions observed during training.

Crossing Markers

In this test, two people cross paths multiple times, at different speeds. Figure 8.4 shows how the system merges the two hats into a single-classified hat when they are connected, while still covering both faces. We are able to accomplish this via the biasses in our transition model, $\mu_{\Delta x}$ and $\mu_{\Delta y}$, which reduces false-negatives when

multiple faces are in a scene. At the following frame in Figure 8.5, the system successfully segments what it determined to be a single hat in the previous frame into two two hats by creating a new Particle Filter.

8.11.2 Construction Site Experiments

The construction site data was collected from footage recorded in March, 2007 at the CITRIS construction site at the University of Berkeley, California, under UCB Human Subjects Protocol #2006-7-1[1]. Because we needed to respect the privacy of the construction workers and restrictions in the protocol, we limited our data acquisition to a one-week period. The video sequence presented contains a number of difficult challenges, particularly partial obstructions of the marker, significant changes in the background due to the tracking of the moving person with a robotic camera, and lighting differences including sharp changes from shadows. Also, the system observes areas of the scene it was not trained on, as the robotic camera moved 10 times to track the construction worker as he walked throughout the construction site. For the construction site, our marker is a green construction vest and we assume the face is located at the top-center of the vest, as we show in Figure 8.1. We first evaluate the performance of the system as we use different color-spaces used for input to AdaBoost. We then evaluate the differences in performance between the Particle Filtered approach from Phase C and the Static Marker Detector from Phase B. All experiments were run on data acquired at 6 fps, simulating that the system can process at this speed, rather than the current capability of 3fps. This diminished recording speed (the max is 10 fps) was caused by requiring us to view the video stream to move the camera to follow a person during recording, while having the system store a secondary video stream to disk for later experimentation. The data suggests that our

[1]To contact UCB Human Subjects, refer to http://cphs.berkeley.edu/content/contact.htm.

system can perform with a 6.0% false positive rate, and 1.2% false negative rate for this real-world application. We summarize our results over a 76 second (331 frame) video sequence from a typical day at the construction site in the following table:

Construction Site Experiments

Experiment	% Correct	FPs	FNs	FP+FNs
Only RGB	19.4%	68.6%	5.1%	6.9%
Only HSV	86.1%	11.5%	1.2%	1.2%
Only LAB	84.3%	10.9%	3.6%	1.2%
All 9 (RGB+HSV+LAB)	93.4%	5.4%	0.6%	0.6%
Static Marker Detector	82.8%	16.3%	0.0%	0.9%
Dynamic Marker Tracker	93.4%	5.4%	0.6%	0.6%

Table 8.2. This tables shows the performance of experiments at the CITRIS construction site. To evaluate the system, we place each frame into the category of correctly obscuring all faces without extraneous ellipses, being a false negative but not false positive, being a false negative but no a false positive, and being both a false negative and false positive. We denote false negatives with FN and false positives with FP.

Color Models

In this test, we investigate how our system performs by using different color spaces, specifically because we are only using simple axis-aligned hyperplanes as our weak hypotheses. We compare the algorithm's performance when just using RGB, just HSV, just LAB, and then the "All 9" dimensional color space of RGB+HSV+LAB. We determined that using all nine is superior in both false positive and false negative rates. This data suggests that color-spaces that explicitly decouple the brightness from the color (LAB and HSV) perform better than those that do not (RGB). This is probably exacerbated by our choice of weak hypotheses that decouple each dimension.

Particle Filtered Data

In this test, we evaluated performance between a non-Particle Filtered approach, where we just use each frame independently as described in Phase B, and using

Figure 8.6. Sample image frame input on left image, with output regions overlayed on right image. This sample illustrates how without Particle Filtering, even with the nine dimensional color-space, partial occlusions segment the visual marker, resulting in multiple small ellipses.

Figure 8.7. Sample image frame input on left image, with output regions overlayed on right image. This sample illustrates how Particle Filtering overcomes partial occlusions, when using the nine dimensional color-space, yielding a single large ellipse.

Particle Filtering as described in Phase C. We can see that the system significantly reduces the number of false-positives from 17.2% to 6.0%, while inducing slightly more false-negatives from 0.9% to 1.2%. There were two extra false-negatives induced by the Particle Filter, one from the shirt being cropped at the bottom of the scene, and one where the previous frame experienced extreme motion blur. We were very strict with our definitions of false-negatives as the face's visible region due to the partially cropped shirt is only 8 pixels wide.

Examples of input and output data can be found in Figures 8.8, 8.9, 8.10, 8.11, and 8.12.

8.12 Machine Learning Discussion

There are a number of other machine learning techniques other than AdaBoost that can be explored. Other machine-learning based classification approaches include K-Nearest Neighbors, Neural Networks and Support Vector Machines. AdaBoost was a good fit to our problem as our formulation has a fairly low dimensional space (our problem only has nine dimensions). Also, as we wanted our system to run in real-time, we wanted a lightweight classification approach. We explicitly chose weak hypotheses that were very fast to evaluate, and axis-aligned hyperplanes require only a lookup for the particular dimension, and a comparison with each selected weak hypothesis. This approach is similar to Viola and Jones' [199] motivation of using simple features that are fast to evaluation. A significant advantage of using the greedy selection method for the next weak hypothesis is that the formulation can be thought of as performing a form of feature selection aswell. If the H dimension in HSV has poor prediction performance, the weak hypotheses associated with H will not be chosen, making the system more robust. This is possible because our weak hypotheses treat each color dimension independently. Feature selection methods such as wrapping [102] have shown to improve classification performance. The motivation of feature reduction approaches follows the principle Occam's Razor. We are searching for the "simplest" way to predict the classification. This AdaBoost formulation implicitly uses feature selection in the step which chooses the next weak hypothesis according to the greedy heuristic in [199]. Exploration of a very high dimensional space (like 10,000 dimensions) is computationally very difficult for our approach, as it would require we explore all weak hypotheses, for each step of the weak hypothesis selection process.

Additionally, we project the RGB space into the HSV and LAB spaces. The original images are provided in RGB, giving us the features in that color-space with no

needed computation. As discussed earlier, HSV is more robust to changing lighting conditions, and LAB is better at handling specularities and is designed to be perceptually linear. As shown in our experiments, using this redundant formulation as well as the implicit feature reduction of AdaBoost, effectively utilizes the additional colorspaces. If all of the best predictive powers were just in the RGB colorspace, then the hyperplanes for HSV and LAB would never be selected. We found the chosen weak hypotheses spanned all of the colorspaces. Projecting into the other two color spaces gives us 3x the features to explore at each step, improving performance, as we show in our experiments. We also chose to use colorspace projections rather than other types of projections as we know they were designed to give robust properties such as attempts at being invariant to lighting conditions.

8.13 Conclusion and Future Work

We have discussed the Respectful Cameras visual privacy system which tracks visual markers to robustly infer the location of individuals wishing to remain anonymous. We discuss a static-image classifier which determines a marker's location using pixel colors and an AdaBoost statistical classifier. We then extended this to marker tracking, using a Particle Filter which uses a Probabilistic AdaBoost algorithm and a marker model which incorporates velocity and interframe information.

It may be possible to build a Respectful Cameras method directly into the camera (akin to the V-chip) so that faces are encrypted at the hardware level and can be decrypted only if a search warrant is obtained.

While a 1.2% false negative rate for the CITRIS construction is encouraging, we would like it to be even lower to better address privacy applications. One encouraging idea is to run the Respectful Cameras system from multiple cameras, and cross

reference the results. Assuming that the false negative rates of multiple cameras is independent, with three cameras, the false-negative rate would be 0.0002%.

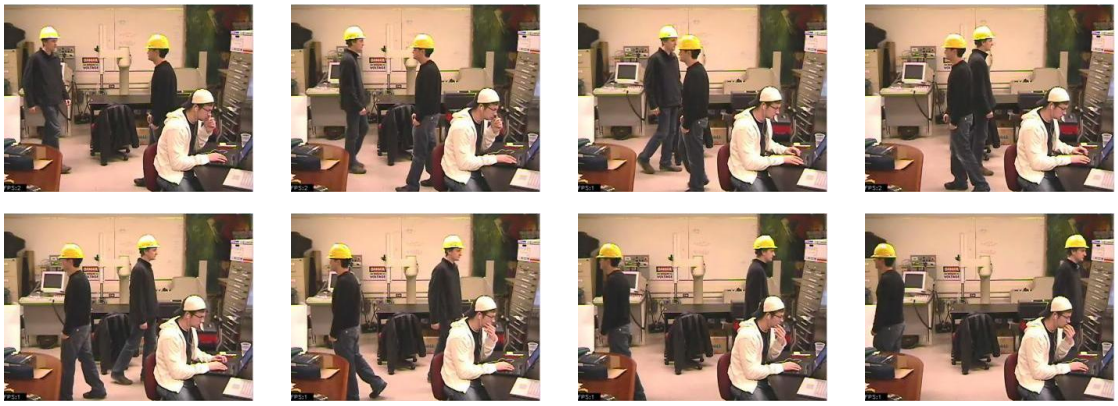


Figure 8.8. Input frames from the in-lab crossing experiment

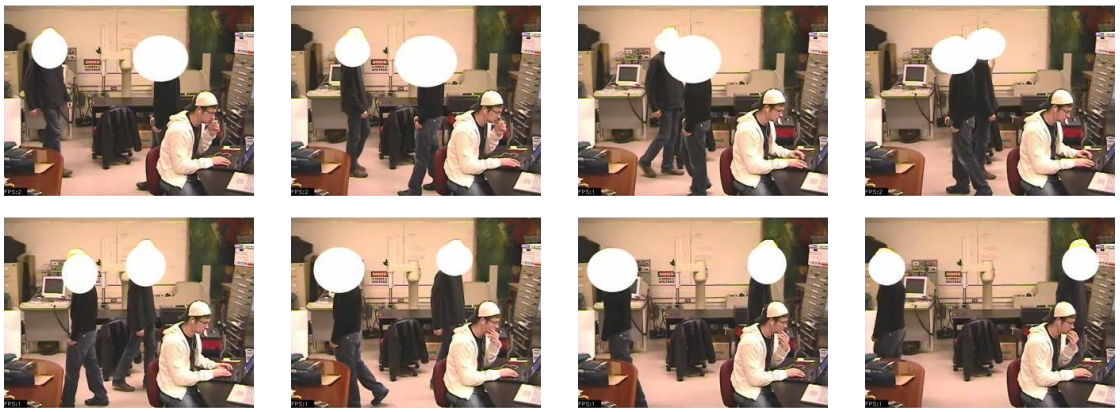


Figure 8.9. Output frames showing the Particle Filter with the nine-dimensional colorspace performs well under dynamic obstructions

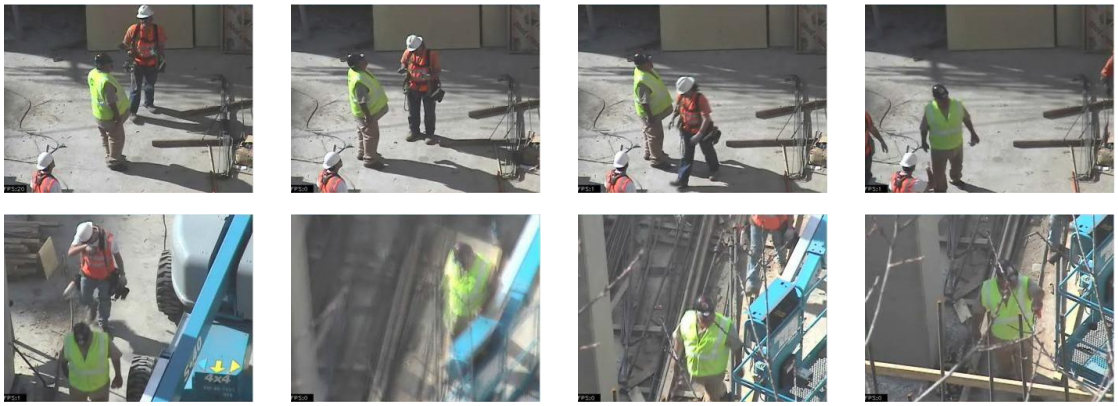


Figure 8.10. Input frames from the CITRIS construction site

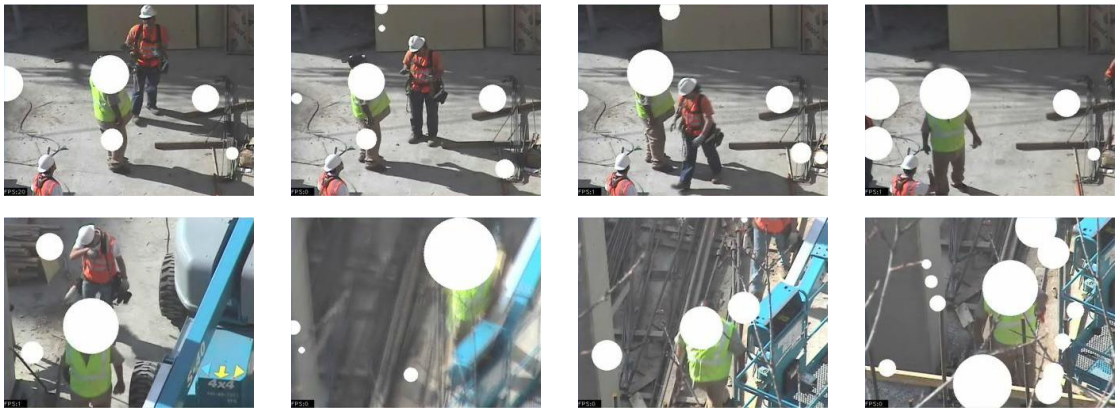


Figure 8.11. Output frames showing using only the RGB colorspace is insufficient

Figure 8.12. Output frames using the full nine-dimension colorspace, showing better performance

Chapter 9

Conclusion And Future Work

In this dissertation, we describe the concept of Structured Tracking, where we can introduce new structure, or leverage existing structure to improve the accuracy and robustness, of tracking problems, while reducing the computational complexity of the algorithms. We show that principled probabilistic inference and machine learning techniques such as Adaptive Boosting, Particle Filtering, and Belief Propagation can improve upon previous methods.

We address the problems of autonomous monitoring using recent technological advances in hardware, distributed algorithms, and machine learning, particularly using sensor networks, camera networks, and robot networks to address societal problems in safety, security, and privacy. For safety, we explore the StatSense Project, where we use spatial correlation between noisy temperature sensors to infer temperatures at unobserved locations. In the Perceptive Pallets project, where we track sensors mounted on pallets to ensure sufficient distance between pallets containing reactive chemicals. For Security, we built the SNARES system, where binary passive infrared sensors tracked the movements of an intruder, and then recorded the intruder's activities with a robotic camera. In Actuator Networks, we explore the algorithms for using

open-loop control of a simple robot by modifying the fields emitted by beacons. For privacy, we develop the Respectful Cameras system, where we track colored markers and use these markers as proxies for people's faces. We then automatically obscure these tracked faces to help preserve their privacy.

This dissertation describes a number of key contributions, but we would like to reiterate the most important.

1. To our knowledge we were the first to implement a Belief Propagation algorithm in a sensor network.

2. We describe an innovative inference algorithm using nonparametric Belief Propagation for real-time tracking of pallets, and using an novel decomposition of the dynamics allowing us to extend a localization approach [86]. When compared to the localization approach, our algorithm has better accuracy, reduced computation, and improved robustness, particularly when pallets are unable to receive at least three inter-distance sensor readings from neighboring pallets. Our approach has 3-4x better accuracy when compared to the SMCL+R method [46]. It is also more robust to noisy connectivity and inter-distance readings because our approach does not assume a unit-disk connectivity model[207].

3. We developed a novel security system which uses a robotic camera and many binary sensors, rather than using many static cameras. We control the camera using a novel Particle Filtering formulation which allowed us to model the spatial response, the object dynamics, and the structure of the environment.

4. We also developed a novel open-loop navigation algorithm which uses robot dynamics and beacons which modify emitted fields such as light or sound to direct a single robot to a goal location.

5. We illustrated how by using colored markers, we can reduce computational com-

plexity and improve tracking robustness to overcome limitations of person and face tracking and show a 2% false negative rate at our CITRIS construction site deployment. We accomplish this with a novel algorithm coupling a probabilistic Adaptive Boosting formulation with a Particle Filter and using a 9-dimensional color space which learns a environment-specific color model.

There are a number of exciting future directions to explore leveraging the insights and methods developed in this dissertation.

9.1 Future Directions

9.1.1 Adding Control to Tracking With Nonparametric Belief Propagation

In the tracking approaches described in this dissertation, we assume the object's movements are observed, but cannot be controlled. However, a major advantage of our Particle Filtering and Belief Propagation approaches is that our inference algorithms do not just provide a value for the most likely state, but incorporate uncertainty models to provide a measure of confidence in our estimate. We can add control into our tracking algorithms, based on our knowledge of this confidence, to more reliably achieve tracking tasks. If we just wish to model the effect the control we have, we can apply standard Kalman Filtering [95] or Particle Filtering [162] techniques. However, if we pose the problem as robustly guiding a robot to a goal location, this becomes a path-planning problem [176, 22, 130, 62]. In some situations, where localization noise varies with location, we need a formulation to address the seemingly antagonistic relationship: at each timestep, do we control each object to move toward a goal location, or do we move the robot to reduce uncertainty?

Two key formulations that model similar theoretical tradeoffs are exploration vs. exploitation [97, 94, 28, 7] and dual control theory [53, 54, 59]. In both of these formulations, the algorithm chooses between control that allows the algorithm to learn more about how control affects the unknown parameters of the system, vs using the estimated parameters to achieve an objective. Note that estimating parameters that affect the system, for instance how a specific control input transitions an object from one state to another, is a slightly different problem from estimating the state of a tracked object.

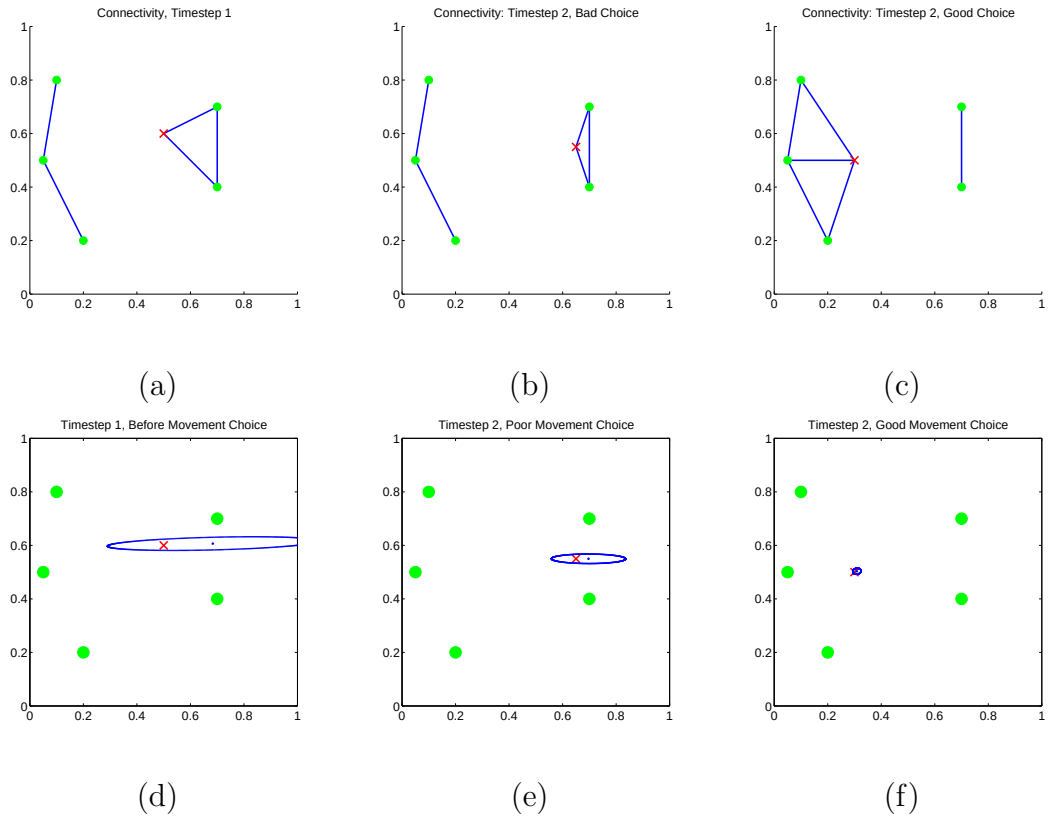


Figure 9.1. Illustration of how different choices of robot control can affect tracking accuracy. Using the same notation as Figure 5.1, subfigures (a)-(c) show the connectivity where (a) shows the first timestep, and (b) and (c) show how different controls yield different positions of the robot at the second timestep. Subfigures (d)-(f) show the localization performance, whereby (e) has poorer localization accuracy than (f) due to the different control choice.

One specific example is in work such as Perceptive Pallets, where the distributions

of inferred states give us a measure of our confidence in the location of each tracked pallet. Thus, for a pallet that is poorly localized, rather than moving it to it's next step to achieve some objective (like being placed on a shelf), as we do not yet have precise enough estimate of the pallet's location, we are better off moving it in a specific way to increase the likelihood that it receives quality inter-sensor distance estimates from enough neighbors, to properly localize itself in the next timestep. Using the Sparse Perceptive Pallets framework, we provide a motivating example in Figure 9.1 illustrating how different control choices can affect the tracking accuracy.

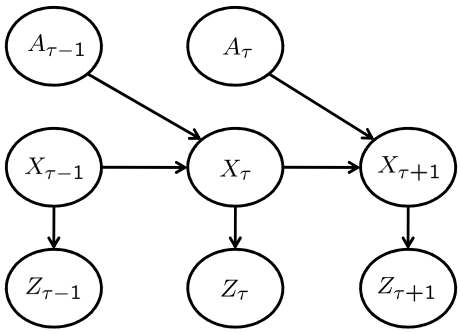


Figure 9.2. The graphical model of a Hidden Markov Model. The graph indicates that we have direct, noisy observations of the state. Also, the next state is determined only by a noisy transition model based only on the previous state and noisy actions applied at the previous timestep.

One method for performing path planning is Markov Decision Processes [19, 186], which assumes transitions between locations are probabilistic, but assume that location of the robot is directly observable (ie. subject to no noise). When we cannot directly observe the location of the robot, as in Perceptive Pallets, and instead have noisy observations for instance from sensor data, we can formulate this problem as a Partially Observable Markov Decision Processes (POMDPs) [6]. Conventional POMDP formulations are over discrete states, but there has been work, for instance by that of Thrun [190], to address continuous states using Monte Carlo methods. They use some of the more recently developed probabilistic machinery, for instance

directly using a Particle Filter in their POMDP observation model formulation to expand beyond discrete models.

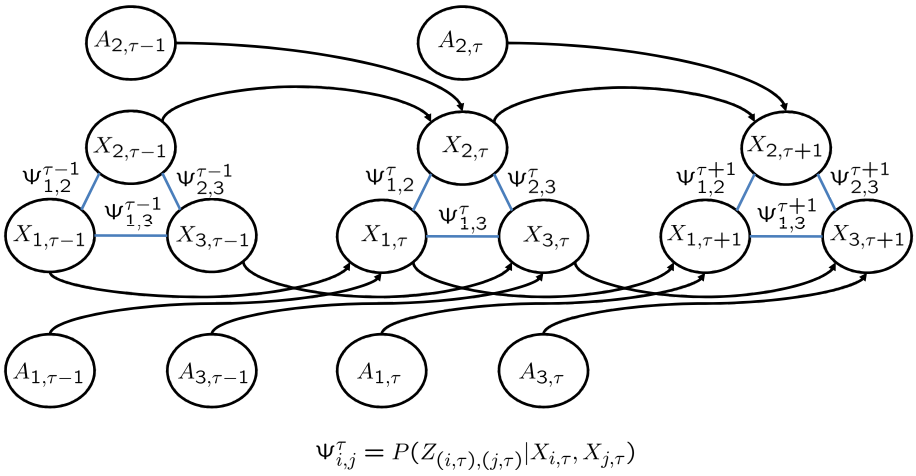


Figure 9.3. This figure illustrates the graphical model for a *Semi-Markovian Model* with an open-loop planner. It is semi-Markovian because each timestep's state depends only on the previous timestep's state. Because the next action does not depend on the previous state, it is an open-loop planner. This representation uses both undirected undirected edges for clarity, where undirected edges correspond to our observation model, and directed correspond to our transition and action models.

We can formulate the types of optimal tracking and control problems we described as POMDPs, but if we would like to apply the ideas to problems such as the Perceptive Pallets described in Chapters 4 and 5, we would like to generalize POMDPs to more complex, semi-Markovian structure. For example, as we can see in Figure 9.2, conventional POMDPs can be described as HMMs, where there are noisy observations of the state, noisy transitions from the state at one timestep to state at the next, and noisy actions which affect the transition. However, we would like to explore how POMDPs can be used for optimal control in environments where observation models are more complicated, requiring models using graphical models with cycles, as depicted in Figures 9.3 and 9.4. This is consistent with the type of observation model presented in Chapter 5 on Sparse Perceptive Pallets. It is important to note that our

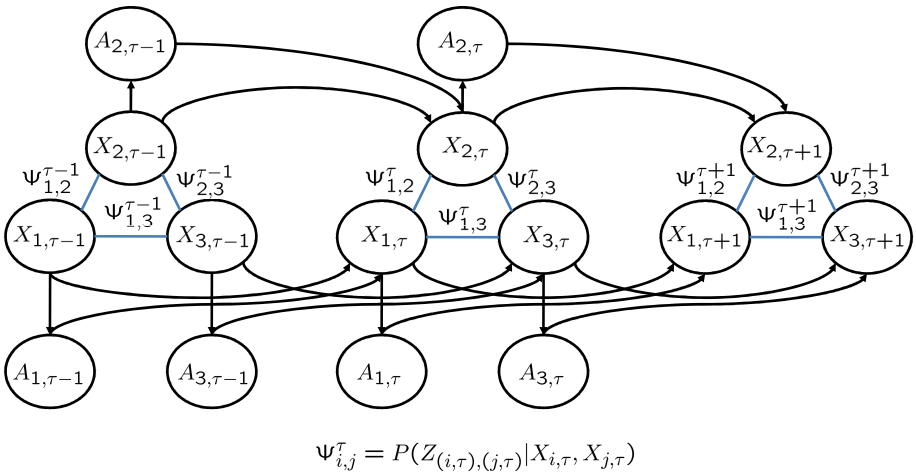


Figure 9.4. This figure illustrates the graphical model for a *Semi-Markovian Model* with a closed-loop planner, where each timestep's state depends only on the previous timestep's state. Because the next action depends on the previous state, it is a closed-loop planner. This representation uses both undirected undirected edges for clarity, where undirected edges correspond to our observation model, and directed correspond to our transition and action models.

formulation is semi-Markovian. By semi-Markovian, we mean that while the observation model has cycles, the transition model only requires knowledge of the previous state allowing us to make use of Markov assumptions. In addition, if we model all objects at a single timestep as a single state vector, then the process can be directly described by an HMM.

We would like to extend Thrun's work on POMDPs, where we can use Belief Propagation (particularly NBP) as our observation model. This could be accomplished by performing NBP for the observation model to determine the distributions of the states of the object, in our case robot locations, and representing this distribution as samples as is done in their more restrictive particle-filtered implementation. We could then use the rest of the framework, with the same reward functions and transition models, thereby adding the ability to control the robot to the problem to our Sparse Perceptive Pallets.

As an alternative approach for solving these noisy path-planning problems, though not formulated for navigation, Grimes et. al [77] has explored a non-parametric inference formulation. This approach is accomplished by defining a graphical model, and modifying the model such that the goal state is set as an observed random variable, and then performing inference. The rationale is that the most likely actions that are determined by inference are those used as control inputs, as they are the most likely to result in the goal state. They use this for learning how to control a robot, and repeatedly uses the best action, as determined by the inference, to iteratively refine learning and control. We could apply these sorts of methods to the variant of Perceptive Pallets for path planning we just described, performing inference on the models illustrated in Figures 9.3 and 9.4 depending on if we have an open, or closed-loop planner.

9.1.2 Identity vs. Position Accuracy in Multi-Object Tracking Problems

There are many scenarios where we have noisy identity information when trying to track the movements of multiple objects. In most tracking frameworks, researchers typically assume one of two extremes regarding identity information. In the first, sensors know exact identity information, as we assume in Perceptive Pallets, because the sensors are placed on the objects that are tracked. Alternatively, other formulations involve sensors that provide spatial information, but no identity information. Instead, other mechanisms for distinguishing the objects like dynamics information must be used [143, 155]. There has been more limited work, particularly in the vision community, that involves uncertain identity information based on models of the objects that are tracked, particularly color models [96, 133]. For instance, Fleuret et al. [60] use background subtraction to determine which locations are people, and then use

color information to determine the most likely mapping between people in one frame to people in another, by examining the consistency of the color models.

We would like to explore how our inference formulations can be extended to also model probabilistic identity information. This would allow for sensors types with different localization vs. identity profiles to be used together to compensate for weaknesses of the other type. For example, we could use a sensor-network and a camera network for tracking pallets in a warehouse. While cameras can be used to precisely localize pallets indoors, we might use color-based tracking but be limited in the number of different colors we could paint the pallets. Thus, the camera-network would have poor identity information because it could not identify the pallet exactly, but could prune out many unlikely pallets it observed. With a sensor on each pallet, as in Perceptive Pallets, we would know exact identity information, as the sensor would know the pallet's identity, but especially in indoor environments, inter-distance localization is poorer than can be achieved by cameras, particularly due to multipath effects. We would like a single formulation that could leverage the identity advantages of the sensor network, and the localization advantages of the cameras to provide a better localization and identity estimate than using these two systems separately. While other mechanisms exist to determine the identity of objects with vision systems such as by using barcodes, other problems such as being able to determine the identity from the barcode on the pallet from any orientation would still need to be addressed. In addition, for domains such as visual tracking of people, we hypothesize a theoretically clean graphical-modeling based inference formulation would perform well with uncertain identity information, even with a single class of sensor.

9.1.3 Deploy NBP Implementation and Alternative Observation Models

As described in Chapter 3 for the StatSense project, we developed a cohesive framework and ran experiments for formulating Belief Propagation on sensor networks using discrete random variables. We have not yet run physical experiments with the NBP framework. We would like to explore extending our StatSense framework and implementation to accommodate NBP while also experimenting with how additional approximations, such as those described in Section 5.4.2 can be used to reduce the computation time of our algorithms to help address the slower processors typically found in sensor-network applications. As NBP can be computationally expensive, we would also like to experiment with hierarchical formulations like in [143], so that a fewer number of high-performance nodes (perhaps in ratios of 20 to 1) can assist with computation, but still in a distributed way. To ease implementation and debugging, declarative sensor-network frameworks like Chu et al. [38] should be explored in addition to directly extending our StatSense Framework. For instance, Singh et al. [177] used the declarative framework for implementing HMMs on sensor networks.

For a physical experiment, any modality with inter-distance readings can be applied, but as we have experience with the MIT cricket motes from the Dense Perceptive Pallets project, that would likely be the type we would use. For our first experiment, we would place 3 beacons in a lab, and then have multiple people hold a mote, and walk around a pre-defined path so that we could compare the system's estimates with ground truth. We would suggest just collecting the data and feeding it into our Java simulator for debugging. Once that was confirmed to work, we would implement a version on the motes and compare accuracy, as well as determine which approximations would be necessary to get this form of NBP running on the motes.

Once these issues were resolved, they could be applied to a real-world application, for instance placing motes on pallets for inventory management.

Depending on the types of information provided by our sensor, we might want to incorporate additional information. For instance, as discussed in the Dense Perceptive Pallets project, there is an angular bias in the inter-distance readings from the cricket motes. We can accomplish this by augmenting the NBP formulation by modifying the observation model in Equations 9.3 and 9.4. If the sensor also provides angular information, we would modify Equation 9.3 to directly use the reading from the sensor, rather than drawing a sample uniformly. For the inter-angle sensor reading Θ_{st} between robot s and robot t, accounting for additive Gaussian error with standard deviation σ_Θ, we would change our model to be:

$$\theta_{st}^{(i)} = \Theta_s + \xi_{st}^{(i)} \qquad \xi_{st}^{(i)} \sim \mathbb{N}\left(0, \sigma_\Theta^2\right) \tag{9.1}$$

$$x_{st}^{(i)} = x_s^{(i)} + (d_{st} + \nu_{st}^{(i)})[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})] \qquad \nu_{st}^{(i)} \sim \mathbb{N}\left(0, \sigma_\nu^2\right) \tag{9.2}$$

If we have a sensor that gives angular information, but no distance information beyond a maximum distance d_{max}, for instance using a led and a camera, we can also modify the observation model accordingly:

$$\theta_{st}^{(i)} = \Theta_s + \xi_{st}^{(i)} \qquad \xi_{st}^{(i)} \sim \mathbb{N}\left(0, \sigma_\Theta^2\right) \tag{9.3}$$

$$x_{st}^{(i)} = x_s^{(i)} + d_{st}[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})] \qquad d_{st}^{(i)} \sim \mathbb{U}\left(0, d_{max}\right) \tag{9.4}$$

Bibliography

[1] "Dust networks," www.dustnetworks.com.

[2] "Panasonic KX-HCM280A." [Online]. Available: http://www2.panasonic.com/consumer-electronics/support/Cameras-Camcorders/Network-Cameras/model.KX-HCM280A

[3] "TRUST: Team for research in ubiquitous secure technology." [Online]. Available: http://www.truststc.org/

[4] "Ubisense: Ultra wide band," http://www.ubisense.net/content/14.html.

[5] "Unblinking: New perspectives on visual privacy in the 21st century." [Online]. Available: http://www.law.berkeley.edu/institutes/bclt/events/unblinking/unblink.html

[6] *AAAI Fall symposium on POMDPs*, 1998. [Online]. Available: http://www.cs.duke.edu/~mlittman/talks/pomdpsymposium.html

[7] P. Abbeel and A. Y. Ng, "Exploration and apprenticeship learning in reinforcement learning," in *In Proceedings of International Conference On Machine Learning (ICML)*, 2005.

[8] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *In Proceedings of Journal on International Journal of Computer Vision (IJCV)*, vol. 1, no. 4, January 1988.

[9] R. Alterovitz, T. Siméon, and K. Goldberg, "The Stochastic Motion Roadmap: A sampling framework for planning with Markov motion uncertainty," in *Proceedings of Conference on Robotics: Science and Systems (RSS)*, June 2007.

[10] A. Amine, S. Ghouzali, and M. Rziza, "Face detection in still color images using skin color information," in *Proceedings of International Symposium on Communications, Control, and Signal Processing (ISCCSP)*, March 13–15 2006.

[11] M. Anderson, "Picture this: Aldermen caught on camera," *Chicago Sun-Times*, Jan., 14 2006.

[12] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on Particle Filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Transactions of Signal Processing*, vol. 50, no. 2, pp. 174–188, February 2002.

[13] S. Avidan, "Spatialboost: Adding spatial reasoning to AdaBoost," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2006, pp. 386–396.

[14] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler, "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," in *IEEE Proceedings of Intelligent Vehicles Symposium*, June 2005, pp. 255–260.

[15] T. Bandyopadhyay, Y. Li, M. Ang Jr., and D. Hsu, "Stealth tracking of an unpredictable target among obstacles," in *Algorithmic Foundations of Robotics VI*, M. Erdmann *et al.*, Eds. Springer-Verlag, 2004, pp. 43–58. [Online]. Available: http://motion.comp.nus.edu.sg/class_rescat.html

[16] B. Barshan and R. Kuc, "A bat-like sonar system for obstacle localization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 636–646, July 1992.

[17] J. Batlle, E. Mouaddib, and J. Salvi, "Recent progress in coded structured light as a technique to solve the correspondence problem: A survey," *Pattern Recognition*, vol. 31, no. 7, pp. 963–982, 1998.

[18] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbocodes," in *International Conference on Communications*, vol. 2, 1993, pp. 1064–1070.

[19] D. P. Bertsekas and J. Tsitsiklis, "Neuro-dynamic programming," in *Athena Scientific*, 1996.

[20] K.-F. Bhringer, V. Bhatt, B. R. Donald, and K. Goldberg, "Algorithms for sensorless manipulation using a vibrating surface," *Algorithmica*, vol. 26, no. 3, pp. 389–429, April 2000.

[21] U. Bischoff, M. Strohbach, M. Hazas, and G. Kortuem, "Constraint-based distance estimation in ad-hoc wireless sensor networks," in *3rd European Workshop on Wireless Sensor Networks (EWSN)*, Zurich, Switzerland, Feb. 2006, pp. 54–68.

[22] L. Blackmore, "A probabilistic particle control approach to optimal, robust predictive control," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2006.

[23] K. F. Böhringer and H. Choset, *Distributed Manipulation*, 1st ed. Kluwer Academic Publishers, 2000.

[24] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms for particle filters: A computational complexity perspective," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 15, pp. 2267–2277, 2004.

[25] B. Bouilly, T. Siméon, and R. Alami, "A numerical technique for planning motion strategies of a mobile robot in presence of uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, Nagoya, Japan, May 1995, pp. 1327–1332.

[26] L. Bourdev and J. Brandt, "Robust object detection via soft cascade," *In Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 236–243, 20-25 June 2005.

[27] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed. O'Reilly, 2008.

[28] R. I. Brafman and M. Tennenholtz, "R-max, a general polynomial time algorithmfor near-optimal reinforcement learning," in *Journal of Machine Learning Research*, 2002.

[29] J. Brassil, "Using mobile communications to assert privacy from video surveillance," *In Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, pp. 8 pp.–, April 2005.

[30] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2000.

[31] M. Cetin, L. Chen, J. W. Fisher, A. T. Ihler, R. L. Moses, M. J. Wainwright, and A. S. Willsky, "Distributed fusion in sensor networks," *IEEE Signal Processing Magazine*, vol. 23, pp. 42–55, 2006.

[32] D. Chen, A. Bharusha, and H. Wactlar, "People identification across ambient camera networks," in *International Conference on Multimedia Ambient Intelligence, Media and Sensing (AIMS)*, April 20 2007.

[33] D. Chen, Y. Chang, R. Yan, and J. Yang, "Tools for protecting the privacy of specific individuals in video," *EURASIP Journal of Applied Signal Processing*, 2007.

[34] L. Chen, M. J. Wainwright, M. Cetin, and A. Willsky, "Multitarget-multisensor data association using the tree-reweighted max-product algorithm," in *SPIE Aerosense Conference*, April 2003.

[35] K. Chinomi, N. Nitta, Y. Ito, and N. Babaguchi, "PriSurv: Privacy protected video surveillance system using adaptive visual abstraction." in *MMM*, ser. Lecture Notes in Computer Science, S. Satoh, F. Nack, and M. Etoh, Eds., vol. 4903. Springer, 2008, pp. 144–154.

[36] H. Choset, K. M. Lynch, S. Hutchingson, G. Kantor, W. Burgand, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, 1st ed. MIT Press, 2005.

[37] H. Choset, K. Nagatani, and N. A. Lazar, "The arc-transversal median algorithm: a geometric approach to increasing ultrasonic sensor azimuth accuracy," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 513–521, 2003.

[38] D. C. Chu, L. Popa, A. Tavakoli, J. M. Hellerstein, P. Levis, S. Shenker, and I. Stoica, "The design and implementation of a declarative sensor network system," in *Embedded Networked Sensor Systems (SenSys)*, November 2007.

[39] C. Connolly, J. Burns, and R. Weiss, "Path planning using Laplace's equation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1990, pp. 2102–2106.

[40] J. M. Coughlan and S. J. Ferreira, "Finding deformable shapes using loopy

belief propagation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, vol. 3, May 2002, pp. 453–468.

[41] C. Crick and A. Pfeffer, "Loopy belief propagation as a basis for communication in sensor networks," in *Uncertainty in Artificial Intelligence*, July 2003.

[42] P. Cudre-Mauroux, M. Fogel, K. Goldberg, and M. Franklin, "Sentry pallets for automated monitoring of spatial constraints: Models and initial experiments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Orlando, Florida, USA, May 2006, pp. 4396–4398.

[43] A. Das, J. Spletzer, V. Kumar, and C. Taylor, "Ad hoc networks for localization and control," in *Proceedings of Conference on Decision and Control*, 2002.

[44] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson, "Planning under time constraints in stochastic domains," *Artificial Intelligence*, vol. 76, no. 1-2, pp. 35–74, July 1995.

[45] J. P. V. den Berg, "A literature survey on planning and control of warehousing systems," *IIE Transactions*, vol. 31, no. 8, pp. 751–762, 1999.

[46] B. Dil, S. Dulman, and P. Havinga, "Range-based localization in mobile sensor networks," in *Proceedings of European Workshop on Wireless Sensor Networks*, 2006, pp. 164–179.

[47] L. Doherty and D. A. Teasdale, "Towards 100% reliability in wireless monitoring networks," in *Proceedings of the Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks (PE-WASUN)*. New York, NY, USA: ACM, 2006, pp. 132–135.

[48] F. Dornaika and J. Ahlberg, "Fast and reliable active appearance model search

for 3-D face tracking," *IEEE Transactions of Systems, Man and Cybernetics, Part B*, vol. 34, no. 4, pp. 1838–1853, 2004.

[49] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, September 2000.

[50] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, June 1996.

[51] J. Fang and G. Qiu, "A colour histogram based approach to human face detection," *International Conference on Visual Information Engineering (VIE)*, pp. 133–136, July 2003.

[52] S. P. Fekete, R. Klein, and A. Nüchter, "Online searching with an autonomous robot," in *Proceedings of Sixth Workshop on Algorithmic Foundations of Robotics*, 2004, pp. 335–350. [Online]. Available: http://www.math.tu-bs.de/~fekete/publications.html

[53] A. A. Fel'dbaum, "Dual control theory, parts i and ii," *Automation and Remote Control*, vol. 21, no. 9, pp. 874–880, April 1961.

[54] ——, "Dual control theory, parts i and ii," *Automation and Remote Control*, vol. 21, no. 11, pp. 1033–1039, April 1961.

[55] P. Felzenszwalb and D. Huttenlocher, "Belief propagation for early vision," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[56] R. Feraud, O. J. Bernier, J.-E. Viallet, and M. Collobert, "A fast and accurate

face detector based on neural networks," *IEEE Transactions of Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 1, pp. 42–53, 2001.

[57] D. Ferguson and A. Stentz, "Focussed dynamic programming: Extensive comparative results," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-04-13, Mar. 2004.

[58] D. A. Fidaleo, H.-A. Nguyen, and M. Trivedi, "The networked sensor tapestry (NeST): a privacy enhanced software architecture for interactive analysis of data in video-sensor networks," in *Proceedings of ACM Workshop on Video Surveillance & Sensor Networks (VSSN)*. New York, NY, USA: ACM Press, 2004, pp. 46–53.

[59] N. Filataov and H. Unbedhauen, *Adaptive Dual Control: Theory and Applications*. Springer, 2004.

[60] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multi-camera people tracking with a probabilistic occupancy map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 267–282, February 2008.

[61] M. Fogel, N. Burkhart, H. Ren, J. Schiff, M. Meng, and K. Goldberg, "Automated tracking of pallets in warehouses: Beacon layout and asymmetric ultrasound observation models," in *Proceedings of Conference on Automation Science and Engineering (CASE)*, Scottsdale, AZ, 2007.

[62] A. Foka and P. Trahanias, "Real-time hierarchical POMDPs for autonomous robot navigation," *Robotics and Autonomous Systems*, vol. 55, no. 7, pp. 561–571, 2007.

[63] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics Principles and Practice*. NY: AW, 1990.

[64] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning lowlevel vision," *International Journal of Computer Vision*, vol. 40, no. 1, pp. 25–47, 2000.

[65] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *In Proceedings of Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[66] B. J. Frey, R. Koetter, and N. Petrovic, "Very loopy belief propagation for unwrapping phase images," in *Neural Information Processing Systems (NIPS)*, 2002, p. 737743.

[67] B. J. Frey and D. J. C. MacKay, "A revolution: Belief propagation in graphs with cycles," in *Neural Information Processing Systems (NIPS)*, 1998, pp. 479–485.

[68] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[69] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proceedings of Information processing in sensor networks (IPSN)*, 2006, pp. 34–42.

[70] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex behavior at scale: An experimental study of low-power wireless sensor networks," UCLA Computer Science, Tech. Rep. 02-0013, 2002.

[71] M. Garofalakis, K. P. Brown, M. J. Franklin, J. M. Hellerstein, D. Z. Wang, E. Michelakis, L. Tancau, E. Wu, S. R. Jeffery, and R. Aipperspach, "Probabilistic data management for pervasive computing: The data furnace project," *IEEE Data Engineering Bulletin*, vol. 29, no. 1, 2006.

[72] R. Gavison, "Privacy and the limits of the law," *89 Yale L.J.*, pp. 421–471, 1980.

[73] J. geun Park, E. D. Demaine, and S. Teller, "Moving-baseline localization," in *Proceedings of Information Processing in Sensor Networks (IPSN)*, 2008, pp. 15–26.

[74] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, no. 2-4, pp. 210–225, 1993.

[75] Google Inc., "Privacy FAQ." [Online]. Available: http://www.google.com/privacy_faq.html#toc-street-view-images

[76] H. Greenspan, J. Goldberger, and I. Eshet, "Mixture model for face-color modeling and segmentation," *Pattern Recognition Letters*, vol. 22, no. 14, pp. 1525–1536, 2001.

[77] D. B. Grimes, R. Chalodhorn, and R. P. N. Rao, "Dynamic imitation in a humanoid robot through nonparametric probabilistic inference." in *Robotics: Science and Systems (RSS)*, G. S. Sukhatme, S. Schaal, W. Burgard, and D. Fox, Eds. The MIT Press, 2006.

[78] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with rfid technology," in *IEEE International Conference on Robotics and Automation (ICRA)*, Boston, Massachusetts, USA, Apr. 2004, pp. 190–202.

[79] A. Hampapur, S. Borger, L. Brown, C. Carlson, J. Connell, M. Lu, A. Senior, V. Reddy, C. Shu, and Y. Tian, "S3: The IBM smart surveillance system: From transactional systems to observational systems," *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–1385–IV–1388, April 2007.

[80] J. Handschin, "Monte Carlo techniques for prediction and filtering of non-linear stochastic processes," *In Proceedings of Journal of Automatica*, vol. 6, pp. 555–563, 1970.

[81] J. L. Hill and D. E. Culler, "Mica: a wireless platform for deeply embedded networks," *Micro, IEEE*, vol. 22, no. 6, pp. 12–24, 2002.

[82] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[83] A. Howard, M. J. Mataric', and G. S. Sukhatme, "Team localization: A maximum likelihood approach," University of Southern California, Tech. Rep., 2002.

[84] L. Hu and D. Evans, "Localization for mobile sensor networks," in *10th annual international conference on Mobile computing and networking*, Philadelphia, Pennsylvania, USA, Sept. 2004, pp. 45–57.

[85] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. John Wiley & Sons, 2001.

[86] A. Ihler, J. Fisher, and A. S. Willsky, "Loopy belief propagation: Convergence and effects of message errors," *Journal of Machine Learning Research*, vol. 6, pp. 905–936, May 2005.

[87] A. Ihler, E. Sudderth, W. Freeman, and A. Willsky, "Efficient multiscale sampling from products of gaussian mixtures," in *Proceedings of Neural Information Processing Systems (NIPS)*, Dec. 2003.

[88] A. Ihler, J. F. III, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 809–819, April 2005.

[89] S. Intille, "Tracking using a local closed-world assumption: Tracking in the football domain," 1994. [Online]. Available: citeseer.ist.psu.edu/intille94tracking.html

[90] V. Isler, S. Kannan, and S. Khanna, "Locating and capturing an evader in a polygonal environment," in *Proceedings of Workshop on Algorithmic Foundations of Robotics*, 2004, pp. 351–367. [Online]. Available: citeseer.ist.psu.edu/isler04locating.html

[91] A. K. Jain, *Fundamentals of digital image processing.* Prentice Hall International, 1989.

[92] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *Proceedings of INFOCOM*, 2004, pp. 2652–2661.

[93] M. I. Jordan, "Graphical models," *Statistical Science*, vol. 19, no. 1, pp. 140–155, 2004.

[94] S. Kakade, M. Kearns, and J. Langford, "Exploration in metric state spaces," in *In Proceedings of International Conference On Machine Learning (ICML)*, 2003.

[95] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the American Society of Mechanical Engineers, Journal of Basic Engineering*, pp. 35–46, March 1960.

[96] J. Kang, I. Cohen, and G. Medioni, "Tracking people in crowded scenes across multiple cameras," in *Asian Conference on Computer Vision*, 2004.

[97] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," in *Machine Learning Journal*, 2002.

[98] O. D. Kellogg, "Foundations of potential theory," 1969.

[99] O. Khatib, "Commande dynamique dans l'espace opérationnel des robots manipulateurs en présence d'obstacles," Ph.D. dissertation, École Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France, 1980.

[100] ——, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[101] J. King, D. K. Mulligan, and S. Raphael, "CITRIS report: The San Francisco community safety camera program," in *CITRIS*, December 2008. [Online]. Available: http://www.citris-uc.org/files/CITRIS%20SF%20CSC%20Study%20Final%20Dec%202008.pdf

[102] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, 1997.

[103] N. Kohtake, J. Rekimoto, and Y. Anzai, "InfoStick: An interaction device for inter-appliance computing," *Lecture Notes in Computer Science*, vol. 1707, pp. 246–258, 1999.

[104] V. Kolmogorov, "Convergent tree-reweighted message-passing for energy minimization," *IEEE Transactions on Pattern Analaysis and Machine Intelligence (PAMI)*, 2006, to appear.

[105] S. G. Kong, J. Heo, B. R. Abidi, J. Paik, and M. A. Abidi, "Recent advances in visual and infrared face recognition: a review," *Transactions of Computer Vision and Image Understanding (CVIU)*, vol. 97, no. 1, pp. 103–135, January 2005.

[106] E. Krotkov and R. Bajcsy, "Active vision for reliable ranging: Cooperating focus, stereo, and vergence," *In Proceeding of International Journal of Computer Vision*, vol. 11, no. 2, pp. 187–203, October 1993.

[107] B. Kusy, J. Sallai, G. Balogh, A. Ledeczi, V. Protopopescu, J. Tolliver, F. DeNap, and M. Parang, "Radio interferometric tracking of mobile wireless nodes," in *5th International Conference on Mobile Systems, Applications, and Services (Mobisys)*, June 2007, pp. 139–150. [Online]. Available: http://www.truststc.org/pubs/269.html

[108] F. Lamiraux and L. E. Kavraki, "Positioning of symmetric and non-symmetric parts using radial and constant fields: Computation of all equilibrium configurations," *International Journal of Robotics Research*, vol. 20, no. 8, pp. 635–659, 2001.

[109] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 43, no. 4, pp. 499–518, 2003.

[110] S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *Journal of the Royal Statistical Society B*, vol. 50, pp. 155–224, January 1988.

[111] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[112] S. M. LaValle and S. A. Hutchinson, "An objective-based framework for motion planning under sensing and control uncertainties," *International Journal of Robotics Research*, vol. 17, no. 1, pp. 19–42, Jan. 1998.

[113] A. Lazanas and J. Latombe, "Motion planning with uncertainty: A landmark approach," *Artificial Intelligence*, vol. 76, no. 1-2, pp. 285–317, 1995.

[114] D.-S. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Transactions of Pattern Analysis and Machine Intelligence*, vol. 27, pp. 827– 832, May 2005.

[115] T. Lefebvre, H. Bruyninckx, and J. D. Schutter, "Kalman Filters for nonlinear systems: a comparison of performance," *International Journal of Control*, vol. 77, no. 7, pp. 639–653, 2004.

[116] Y. Lei, X. Ding, and S. Wang, "AdaBoost tracker embedded in adaptive Particle Filtering," in *Proceedings of International Conference on Pattern Recognition (ICPR)*, vol. 4, Aug. 2006, pp. 939–943.

[117] A. Levin and Y. Weiss, "Learning to combine bottom-up and top-down segmentation," in *European Conference on Computer Vision (ECCV)*, June 2006.

[118] Q. Li, M. D. Rosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *Proceedings of Mobile Computing and Networking (MobiCom)*. New York, NY, USA: ACM Press, 2003, pp. 313–325.

[119] H.-J. Lin, S.-H. Yen, J.-P. Yeh, and M.-J. Lin, "Face detection based on skin color segmentation and SVM classification," *Secure System Integration and Reliability Improvement (SSIRI)*, pp. 230–231, 2008.

[120] E. C. Liu and J. M. F. Moura, "Fusion in sensor networks: convergence study," in *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, vol. 3, May 2004, pp. 865–868.

[121] K. Lorincz and M. Welsh, "Motetrack: a robust, decentralized approach to rf-based location tracking," *Personal Ubiquitous Comput.*, vol. 11, no. 6, pp. 489–503, 2007.

[122] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," in *International Journal of Computer Vision*, vol. 20, 2003, pp. 91–110.

[123] M. Maróti, P. Völgyesi, S. Dóra, B. Kusy, A. Nádas, Á. Lédeczi, G. Balogh, and K. Molnár, "Radio interferometric geolocation," in *Proceedings of the conference on Embedded networked sensor systems (SenSys)*, 2005, pp. 1–12.

[124] D. P. Massa, "Choosing an ultrasonic sensor for proximity or distance measurement; part 2: Optimizing sensor selection," *Sensors*, vol. 16, no. 3, pp. 28–43, 1999.

[125] J. C. Maxwell, "On hills and dales," *The Philosophical Magazine*, vol. 40, no. 269, pp. 421–427, 1870.

[126] M. McCahill and C. Norris, "From cameras to control rooms: the mediation of the image by CCTV operatives," *CCTV and Social Control: The politics and practice of video surviellance-European and global perspectives*, 2004.

[127] R. J. McEliece, D. J. C. MacKay, and J. Cheng, "Turbo decoding as an instance of pearls "belief propagation" algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140–152, February 1998.

[128] A. Meliou, D. Chu, C. Guestrin, J. Hellerstein, and W. Hong, "Data gathering tours in sensor networks," in *Proceedings on Information Processing in Sensor Networks (IPSN)*, April 2006.

[129] T. Meltzer, C. Yanover, and Y. Weiss, "Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation," in *International Conference on Computer Vision (ICCV)*, June 2005.

[130] J.-A. Meyer and D. Filliat, "Map-based navigation in mobile robots:: Ii. a

review of map-learning and path-planning strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 283–317, 2003.

[131] K. Michael and L. McCathie, "The pros and cons of rfid in supply chain management," in *International Conference on Mobile Business (ICMB '05)*, Sydney, Australia, July 2005, pp. 623–629.

[132] A. Micilotta and R. Bowden, "View-based location and tracking of body parts for visual interaction." in *Proceedings of British Machine Vision Conference*, vol. 2, September 2004, pp. 849–858. [Online]. Available: http://www.amicilotta.airpost.net/

[133] A. Mittal and L. S. Davis, "M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene," *International Journal of Computer Vision*, vol. 51, no. 3, pp. 189–203, February 2003.

[134] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *2nd international conference on Embedded networked sensor systems*, Baltimore, Maryland, USA, Nov. 2004, pp. 50–61.

[135] M. T. Moore, "Cities opening more video surveillance eyes," *USA Today*, July 18, 2005.

[136] O. L. Moses and R. Patterson, "Self-calibration of sensor networks," in *Proceedings of SPIE on Unattended Ground Sensor Technologies and Applications IV*, vol. 4743, Orlando, FL, April 2002.

[137] J. M. F. Moura, J. Lu, and M. Kleiner, "Intelligent sensor fusion: a graphical model approach," in *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, April 2003.

[138] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of Uncertainty in AI*, 1999, pp. 467–475. [Online]. Available: citeseer.ist.psu.edu/murphy99loopy.html

[139] New York Civil Liberties Union (NYCLU), "Report documents rapid proliferation of video surveillance cameras, calls for public oversight to prevent abuses," December 13 2006. [Online]. Available: http://www.nyclu.org/whoswatching_pr_121306.html

[140] W. S. Newman and N. Hogan, "High speed robot control and obstacle avoidance using dynamic potential functions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1987, pp. 14–24.

[141] H. F. Nissenbaum, "Privacy as contextual integrity," *Washington Law Review*, vol. 79, no. 1, 2004.

[142] S. Oh and S. Sastry, "An efficient algorithm for tracking multiple maneuvering targets," in *Proceedings of International Conference on Decision and Control*, Seville, Spain, December 2005.

[143] S. Oh, L. Schenato, P. Chen, and S. Sastry, "Tracking and coordination of multiple agents using sensor networks: system design, algorithms and experiments," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 234–254, January 2007.

[144] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe, "A boosted Particle Filter: Multitarget detection and tracking," in *Proceedings of European Conference on Computer Vision (ECCV)*, May 2004, pp. 28–39.

[145] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer, "Weak hypotheses and boosting for generic object detection and recognition," in *Proceedings of Conference European Conference on Computer Vision (ECCV)*, 2004, pp. 71–84.

[146] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," *Proceedings of IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, pp. 130–136, 1997.

[147] M. Paskin, C. Guestrin, and J. McFadden, "A robust architecture for distributed inference in sensor networks," in *Proceedings on Information Processing in Sensor Networks (IPSN)*, April 2005.

[148] J. Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[149] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2002, pp. 661–675.

[150] L. C. A. Pimenta, G. A. S. Pereira, and R. C. Mesquita, "Fully continuous vector fields for mobile robot navigation on sequences of discrete triangular regions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1992–1997.

[151] J. Polastare, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *4th international symposium on Information processing in sensor networks*, Los Angeles, California, USA, Apr. 2005, p. 48.

[152] N. Priyantha, A. K. Miu, H. Balakrishnan, and S. Teller, "The cricket compass for context-aware mobile applications," in *International Conference on Mobile Computing and Networking*, Rome, Italy, July 2001, pp. 1–14.

[153] N. B. Priyantha, "The cricket indoor location system," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 2005.

[154] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proceedings of the Conference on Mobile Computing and Networking (MOBICOM)*, Boston, MA, August 2000.

[155] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, pp. 843–854, December 1979. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1102177

[156] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.

[157] I. Rish, "Distributed systems diagnosis using belief propagation," in *Allerton Conference on Communication, Control, and Computing*, October 2005.

[158] C. Rohrig and S. Spieker, "Tracking of transport vehicles for warehouse management using a wireless sensor network," in *Proceedings of Intelligent Robots and Systems (IROS)*, September 2008, pp. 3260–3265.

[159] T. Roosta, M. J. Wainwright, and S. Sastry, "Convergence analysis of reweighted sum-product algorithms," in *Proceedings on International Conference Acoustic, Speech and Signal Processing*, April 2007.

[160] A. Rosenfeld, "Connectivity in digital pictures," *J. ACM*, vol. 17, no. 1, pp. 146–160, 1970.

[161] S. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, Oct 2002.

[162] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Pearson Education, 1995.

[163] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *ACM Workshop on Wireless Sensor Networks and Applications*, 2003, pp. 112–121.

[164] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Computational Learing Theory*, pp. 80–91, 1998.

[165] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," *Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 195–202, June 2003.

[166] J. Schiff and K. Goldberg, "Automated intruder tracking using particle filtering and a network of binary motion sensors," in *IEEE International Conference on Automation Science and Engineering (CASE)*, Shanghai, China, Oct. 2006, pp. 1–2.

[167] ——, "Automated intruder tracking using particle filtering and a network of binary motion sensors," *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 580–587, 8-10 Oct. 2006.

[168] J. Schiff, M. Meingast, D. K. Mulligan, S. Sastry, and K. Goldberg, "Respectful cameras: detecting visual markers in real-time to address privacy concerns," *In Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, pp. 971–978, Oct. 29 2007-Nov. 2 2007.

[169] A. Senior, A. Hampapur, Y. Tian, L. Brown, S. Pankanti, and R. Bolle, "Appearance models for occlusion handling," *Journal of Image and Vision Computing (IVC)*, vol. 24, no. 11, pp. 1233–1243, November 2006.

[170] A. Senior, S. Pankanti, A. Hampapur, L. Brown, Y.-L. Tian, A. Ekin, J. Connell, C. F. Shu, and M. Lu, "Enabling video privacy through computer vision," *IEEE Security & Privacy*, vol. 3, no. 3, pp. 50–57, May-June 2005.

[171] Y. Shang, W. Ruml, Y. Zhang, and M. P. Fromherz, "Localization from mere connectivity," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing*, 2003, pp. 201–212.

[172] R. Shaw, "Recognition markets and visual privacy," in *UnBlinking: New Perspectives on Visual Privacy in the 21st Century*, November 2006.

[173] C. Shen, B. Wang, F. Vogt, S. Oldridge, and S. Fels, "Remoteeyes: A remote low-cost position sensing infrastructure for ubiquitous computing," in *Proceedings of International Workshop on Networked Sensing Systems*, Japan, June 2004, pp. 31–35.

[174] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM national conference.* New York, NY, USA: ACM Press, 1968, pp. 517–524.

[175] B. Silverman, "Density estimation for statistics and data analysis," *New York: Chapman and Hall*, 1986.

[176] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995, p. 10801087.

[177] A. Singh, C. R. Ramakrishnan, D. S. W. I V. Ramakrishnan, and J. L. Wong, "A methodology for in-network evaluation of integrated logical-statistical models," in *Embedded Networked Sensor Systems (SenSys 2008)*, 2008, pp. 197–210.

[178] A. Smith, H. Balakrishnan, M. Goraczko, and N. B. Priyantha, "Tracking Moving Devices with the Cricket Location System," in *Proceedings of Conference on Mobile Systems, Applications and Services (Mobisys)*, Boston, MA, June 2004.

[179] B. Sohn, J. Lee, H. Chae, and W. Yu, "Localization system for mobile robot using wireless communication with IR landmark," in *Proceedings of Robot Communication and Coordination (ROBOCOMM)*, 2007, p. Article No. 6.

[180] D. Song, "Systems and algorithms for collaborative tele-operation," Ph.D. dissertation, University of California, Berkeley, 2004.

[181] E. Steltz, S. Avadhanula, and R. S. Fearing, "High lift force with 275 hz wing beat in mfi," in *International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 3987–3992.

[182] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, "A high-accuracy, low-cost localization system for wireless sensor networks," in *3rd international conference on Embedded networked sensor systems*, San Diego, California, Nov. 2005, pp. 13–26.

[183] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2003.

[184] A. Sudsang and L. Kavraki, "A geometric approach to designing a programmable force field with a unique stable equilibrium for parts in the plane," in *Proceedings of the IEEE Interational Conference on Robotics and Automation*, vol. 2, 2001, pp. 1079–1085.

[185] J. Sun, H. Shum, and N. Zheng, "Stereo matching using belief propagation," in *European Conference on Computer Vision*, 2002, pp. 510–524.

[186] R. S. Sutton and A. G. Barto, "Reinforcement learning," in *MIT Press*, 1998.

[187] M. J. Swain and M. A. Stricker, "Promising directions in active vision," *In*

Proceeding of International Journal of Computer Vision, vol. 11, pp. 109–126, 1993.

[188] S. Tatikonda and M. I. Jordan, "Loopy belief propagation and Gibbs measures," in *Proceedings on Uncertainty in Artificial Intelligence*, vol. 18, August 2002, pp. 493–500.

[189] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *Proceedings of Information processing in sensor networks (IPSN)*, April 2006, pp. 27–33.

[190] S. Thrun, "Monte carlo POMDPs," in *Advances in Neural Information Processing Systems 12*, S. Solla, T. Leen, and K.-R. Mller, Eds. MIT Press, 2000, pp. 1064–1070.

[191] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 1st ed. MIT Press, 2005.

[192] S. Thrun and Y. Liu, "Multi-robot SLAM with sparse extended information filers," in *Proceedings of International Symposium of Robotics Research (ISRR)*. Springer, 2003.

[193] S. Thrun, "Probabilistic robotics," *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[194] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, Massachusetts: MIT Press, 2005.

[195] S. Thrun and Y. Liu, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, July-August 2004.

[196] M. Turk and A. Pentland, "Face recognition using Eigenfaces," in *Proceedings of IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, 1991, pp. 586–591.

[197] N. Ukita and T. Matsuyama, "Real-time cooperative multi-target tracking by communicating active vision agents," *In Proceedings of Journal on Computer Vision and Image Understanding*, vol. 97, no. 2, pp. 137–179, February 2005.

[198] P. Viola and M. Jones, "Robust real-time face detection," *In Proceedings of International Journal of Computer Vision (IJCV)*, vol. 57, no. 2, pp. 137–154, May 2004.

[199] ——, "Rapid object detection using a boosted cascade of simple features," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 01, p. 511, 2001.

[200] J. von Neumann, "Various techniques used in connection with random digits," *Applied Math Series*, vol. 12, pp. 36–38, 1951.

[201] T. H. Vose, P. Umbanhowar, and K. M. Lynch, "Vibration-induced frictional force fields on a rigid plate," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2007, pp. 660–667.

[202] M. J. Wainwright, "Estimating the "wrong" graphical model: Benefits in the computation-limited regime," *Journal of Machine Learning Research*, pp. 1829–1859, September 2006.

[203] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Exact MAP estimates via agreement on (hyper)trees: Linear programming and message-passing," *IEEE Transactions on Information Theory*, vol. 51, no. 11, pp. 3697–3717, November 2005.

[204] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. M. Hellerstein, "Bayesstore: Managing large, uncertain data repositories with probabilistic graphical models," in *Very Large Data Bases (VLDB)*, Auckland, New Zealand, 2008.

[205] X. Wang and S. Wang, "Collaborative signal processing for target tracking in distributed wireless sensor networks," *In Proceedings of Journal of Parallel and Distributed Computing*, vol. 67, no. 5, May 2007.

[206] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *In Proceedings of Journal of Neural Computation*, vol. 12, no. 1, pp. 1–41, 2000.

[207] K. Whitehouse, "Understanding the prediction gap in multi-hop localization," Ph.D. dissertation, University of California at Berkeley, 2006.

[208] W. Wiegerinck, "Approximations with reweighted generalized belief propagation," in *Workshop on Artificial Intelligence and Statistics*, January 2005.

[209] J. Wilson, V. Bhargava, A. Redfern, and P. Wright, "A wireless sensor network and incident command interface for urban firefighting," in *Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous*, Aug. 2007, pp. 1–7.

[210] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997. [Online]. Available: citeseer.ist.psu.edu/wren97pfinder.html

[211] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors," *IEEE International Conference on Computer Vision (ICCV)*, vol. 1, pp. 90–97, 2005.

[212] ——, "Tracking of multiple, partially occluded humans based on static body part detection," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA, 2006.

[213] Y.-W. Wu and X.-Y. Ai, "Face detection in color images using AdaBoost algorithm based on skin color information," in *International Workshop on Knowledge Discovery and Data Mining (WKDD)*, 2008, pp. 339–342.

[214] M. Yang, D. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions of Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 1, pp. 34–58, January 2002.

[215] P. Yang, R. A. Freeman, and K. M. Lynch, "Distributed cooperative active sensing using consensus filters," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, April 2007, pp. 405–410.

[216] P. Yang, R. Freeman, and K. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, December 2008.

[217] C. Yanover, T. Meltzer, and Y. Weiss, "Linear programming relaxations and belief propagation: An empirical study," *Journal of Machine Learning Research*, vol. 7, pp. 1887–1907, September 2006.

[218] C. Yanover and Y. Weiss, "Approximate inference and proteinfolding," in *Neural Information Processing Systems (NIPS)*, 2003, pp. 1457–1464.

[219] J. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms," *IEEE Transactions on Information Theory*, vol. 51, no. 7, pp. 2282–2312, July 2005.

[220] W. Zhang, S. ching S. Cheung, and M. Chen, "Hiding privacy information in video surveillance system," *In Proceedings of IEEE International Conference on Image Processing (ICIP)*, vol. 3, pp. II–868–71, Sept. 2005.

[221] X. Zhang, S. Fronz, and N. Navab, "Visual marker detection and decoding in AR systems: a comparative study," in *International Symposium on Mixed and Augmented Reality*, 2002, pp. 97–106.

[222] F. Zhao, J. Liu, J. Liu, L. Guibas, and J. Reich, "Collaborative Signal and Information Processing: an Information-Directed Approach," *IEEE Transactions on Communication*, vol. 91, no. 8, pp. 1199–1209, August 2003.